版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

CETC 云计算技术与应用专业校企合作系列教材

Java Web 云应用开发 项目式教程

主　编　罗　颖　伊雯雯
副主编　汤晓燕　胡　霞
　　　　刘文军　王　喜

高等教育出版社

CETC 云计算技术与应用专业校企合作系列教材

Java Web 云应用开发
项目式教程

主　编　罗　颖　伊雯雯
副主编　汤晓燕　胡　霞
　　　　刘文军　王　喜

高等教育出版社·北京

内容简介

本书为云计算技术与应用专业校企合作教材。

本书较为系统地介绍了使用 Java Web 云应用开发项目的相关知识和技能，依托云服务器平台，全面讲解了云应用开发和部署过程，涉及的技术包括基于 Spring+SpringMVC+MyBatis 框架开发、Ajax 异步通信技术以及各种数据可视化相关技术等。

本书以新能源汽车智能监控管理系统的开发为主线，采用项目式教学，结合当前主流开发技术，并根据系统中每个模块的特点和教学侧重点不同，将其科学地划分为基础知识篇、技能训练篇和云平台部署发布篇 3 篇，并根据 Java Web 开发程序员的岗位要求和职业标准，系统地分为 10 个项目，涵盖了 Java Web 云应用开发程序员岗位的基本知识与技能要求。

本书可作为高等职业及应用型本科院校云计算技术与应用专业、软件技术专业的核心课程以及计算机相关专业的教材，也可作为软件技术应用和云计算技术与应用的培训教材，同时还可供设计人员参考使用。

图书在版编目（C I P）数据

Java Web 云应用开发项目式教程 / 罗颖，伊雯雯主编．--北京：高等教育出版社，2018. 8
ISBN 978-7-04-049836-3

Ⅰ. ①J…　Ⅱ. ①罗…　②伊…　Ⅲ. ①JAVA 语言-程序设计-高等职业教育-教材　Ⅳ. ①TP312. 8

中国版本图书馆 CIP 数据核字(2018)第 107061 号

策划编辑　吴鸣飞　　责任编辑　吴鸣飞　　封面设计　赵　阳　　版式设计　徐艳妮
插图绘制　于　博　　责任校对　胡美萍　　责任印制　耿　轩

出版发行　高等教育出版社
社　　址　北京市西城区德外大街 4 号
邮政编码　100120
印　　刷　北京七色印务有限公司
开　　本　787mm×1092mm　1/16
印　　张　19. 25
字　　数　510 千字
购书热线　010-58581118
咨询电话　400-810-0598

网　　址　http://www.hep.edu.cn
　　　　　http://www.hep.com.cn
网上订购　http://www.hepmall.com.cn
　　　　　http://www.hepmall.com
　　　　　http://www.hepmall.cn
版　　次　2018年 8 月第 1 版
印　　次　2018年 8 月第 1 次印刷
定　　价　43. 60 元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

物 料 号　49836-00

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

前　言

近年来，云计算技术的迅猛发展给 IT 行业带来了新鲜的活力，云计算技术的兴起给很多企业带来了蓬勃发展的机会，越来越多的企业开始关注并选择基于云计算的软件或者硬件服务。人才市场的巨大需求对云计算开发工程师、Web 应用程序开发工程师提出了更高的要求，其不仅需要掌握基本的 Java Web 应用开发技术，同时还需要熟悉云平台搭建、部署，以及大数据分析的相关技术。为适应高等院校对云计算技术与应用专业教学的需求，校企合作将陆续推出系列专业教材，本书就是在此背景下，由成员单位苏州工业职业技术学院、南京第五十五所技术发展有限公司和江苏一道云科技发展有限公司共同编写而成的，是校企产教融合后的实践产物。本书结合 Java Web 应用开发、数据可视化技术、云计算平台搭建与部署作为核心内容，是针对高等院校云计算技术与应用专业或相关专业的云应用与开发课程的参考教材。本书以新能源汽车智能监控管理系统的开发为主线，将新能源汽车项目的开发分解为若干子模块，在每个子模块中，首先提出要完成的任务目标，然后详细描述任务的实现步骤，并对项目涉及的技术原理进行详细介绍，最后给出每个模块的实现效果。本书在基础知识介绍具有针对性，任务目标操作具体化。新能源汽车智能监控管理系统使用业内主流的 Java Web 框架开发技术、数据可视化技术，结合云平台的搭建和部署，项目功能全面、系统扩展性强，具有良好的实用性。

1. 特点

（1）项目贯穿

本书以一个 Java Web 云应用系统——新能源汽车智能监控管理系统为开发主线，贯穿全书 3 篇——基础知识篇、技能训练篇和云平台部署发布篇，并根据 Java Web 开发程序员的岗位要求和职业标准，系统地分为 10 个项目，涵盖了 Java Web 云应用开发程序员岗位的基本知识与技能要求。

（2）任务驱动

10 个项目各包含多个功能任务，在基础知识篇中，每个任务的内容按实际教学实施的步骤分为任务目标和技术要点，并在每个项目最后进行项目总结；在技能训练篇和云平台部署发布篇中，每个项目开始会罗列对整个项目的项目描述、知识目标、技能目标以及项目所包含的任务列表 4 部分内容。每个任务的内容按照实际教学实施的步骤分为任务描述、任务目标、任务分析、任务实施 4 个环节，在项目结束时有一个技能训练环节，考查学生的学习情况，最后再对每个项目进行总结。任务设计的思路是：先“做中学”，通过完成功能掌握技能，学习知识；再“学中做”，通过掌握的技能与知识进行进一步的技能训练与拓展。

（3）技能点与知识点紧密融合

每个任务的完成都需要知识点与技能点的支撑，学生在完成任务的同时也实现了对知识与技能点的消化，知识点与技能点已经融合在任务中，不可分割。

2. 内容

本书案例项目采用的是校企产教融合的实践项目——新能源汽车智能监控管理系统，共分为基础知识篇、技能训练篇与云平台部署发布篇 3 篇。

（1）基础知识篇

项目 1 云应用概述，介绍 Java Web 开发、云计算等相关知识，包括 Java Web 开发框架、云计算概念、云服务、云应用等内容，使读者对 Java Web 云应用开发形成宏观的认知，有利于后续内容的学习。

项目 2 开发技术概述，介绍系统开发过程中用到的关键技术和框架，包括 Bootstrap 网页框架、jQuery 实现 Ajax 异步通信技术、Spring 框架、SpringMVC 框架、MyBatis 框架以及 E-Charts 图表可视化插件。

项目 3 开发环境与工具介绍，介绍系统开发过程中开发环境中用到的工具软件。

（2）技能训练篇

项目 4 系统概要设计，介绍系统的需求分析、详细设计以及数据库设计。概要设计和数据库设计为后续的任务实施规划了方案，是任务实施的关键环节。

项目 5 构建本地开发环境，介绍安装配置 JDK、Tomcat、Eclipse、MySQL 的详细过程，同时讲解了第一个 Java Web 项目的创建、部署与运行过程。

项目 6 会员信息管理模块，通过实现会员登录、注册、信息显示以及信息更新等功能，详细讲解基于 SSM 框架的 Java Web 开发过程和方法。

项目 7 车辆信息管理模块，实现车辆信息的管理，包括车辆新增、批量导入、分页、查询、更新等功能。

项目 8 车辆信息监控模块，实现车辆信息的监控，包括分布地图展示、车辆按条件查询实现、故障车辆地图展示、车辆按省份地图展示、指定车辆数据展示、车辆按省份统计分析等功能。

项目 9 车友圈模块，实现添加车友、车友圈信息发布、内容展示、点赞评论等功能。

（3）云平台部署发布篇

项目 10 云平台部署发布，详细介绍 Linux 虚拟机的安装配置、Docker 容器的安装与配置、Tomcat 和 MySQL 镜像的拉取与安装、项目工程部署、制作软件镜像。

3. 使用说明

（1）教学内容课时安排

本书建议授课 96 学时，教学单元与课时安排见表 1。

表 1　教学单元与课时安排

序　号	项 目 名 称	学时安排课时	篇章安排课时
1	云应用概述	2	基础知识篇 8
2	开发技术概述	4	
3	开发环境与工具介绍	2	
4	系统概要设计	2	技能训练篇 78
5	构建本地开发环境	2	
6	会员信息管理模块	18	
7	车辆信息管理模块	20	
8	车辆信息监控模块	20	
9	车友圈模块	16	
10	云平台部署发布	10	云平台部署发布篇 10
课时总计		96	

续表

（2）课程资源一览表

本书是云计算技术与应用专业校企合作系列教材，其同时开发了丰富的数字化教学资源，可使用的教学资源见表 2。

表 2　课程教学资源一览表

序　号	资 源 名 称	表现形式与内涵
1	课程简介	Word 文档，包括对课程内容的简单介绍和对课时、适用对象等项目的介绍，让学习者对 Java Web 云应用开发有个简单的认识
2	学习指南	Word 文档，包括对学前要求、学习目标要求以及学习路径和考核标准要求，让学生知道如何使用资源完成学习
3	课程标准	Word 文档，包括课程定位、课程目标要求以及课程内容与要求，可供教师类读者备课时使用
4	教案文档	Word 文档，包括课程设计思路，课程具体的目标要求以及课程内容设计和能力训练设计，同时给出考核方案设计，让教师理解课程的设计理念，有助于教学实施
5	课件 PPT 和视频	PPT 文件和 MP4 视频文件，可帮助教师、学生更好理解授课内容
6	案例	Tar 包，包括单元项目案例和综合案例，要求综合运用所学知识
7	习题库、试卷库	Word 文档，习题包括理论习题和操作习题，试卷包括单元测试和课程测试。通过练习和测试，让学习者加深对知识的掌握程度
8	附书源码	Tar 包，包括本书中所有例题和任务的源代码

本书配套的教学资源请读者登录 http://www.1daoyun.com 下载或发送邮件至 1548103297@qq.com 索取。

4. 致谢

本书由苏州工业职业技术学院伊雯雯等老师、南京第五十五所技术开发有限公司和江苏一道云科技发展有限公司合作共同编写。感谢陈晓和袁婉敏在本书编写过程中给予的技术支持。

由于作者水平有限，错误和不足之处在所难免，恳请各位读者给予批评、指正，将不胜感激。

编者
2018. 4

版权相关注意事项：
1、书籍版权归著者和出版社所有
2、本PDF来自于各个广泛的信息平台，经过整理而成
3、本PDF仅限用于非商业用途或者个人交流研究学习使用
4、本PDF获得者不得在互联网上以任何目的进行传播，违规者造成的法律责任和后果，违规者自负
5、如果觉得书籍内容很赞，请一定购买正版实体书，多多支持编写高质量的图书的作者和相应的出版社！当然，如果图书内容不堪入目，质量低下，你也可以选择狠狠滴撕裂本PDF
6、技术类书籍是拿来获取知识的，不是拿来收藏的，你得到了书籍不意味着你得到了知识，所以请不要得到书籍后就觉得沾沾自喜，要经常翻阅！！经常翻阅
7、请于下载PDF后24小时内研究使用并删掉本PDF

目　录

第 1 篇　基础知识篇

第 2 篇　技能训练篇

第 3 篇 云平台部署发布篇

第1篇 基础知识篇

项目1 云应用概述

PPT 云应用概述

项目描述

本项目涉及云应用开发所涉及的相关知识。 通过本项目内容的学习，了解云计算的相关知识，包括云计算、云服务、云应用；了解 Docker 容器、Docker 的组织架构以及 Docker 和虚拟机的对比。

知识目标

- 了解云计算、云服务及云应用。
- 了解 Docker 容器的基本组织架构、Docker 容器与虚拟机的对比。

任务列表

任务编号	任务名称	建议课时
任务 1.1 任务 1.2	认识云计算 认识 Docker	
	总计：	2 课时

任务 1.1　认识云计算

【任务目标】

- 了解云计算概念。
- 了解云服务概念。
- 了解云应用概念。

微课 1-1
认识云计算

【技术要点】

1. 云计算概述

云计算（Cloud Computing）是一种按使用量付费的模式，这种模式提供可用的、便捷的、按需的网络访问，进入可配置的计算资源共享池（资源包括网络、服务器、存储、应用软件、服务），这些资源能够被快速提供，只需投入很少的管理工作，或与服务供应商进行很少的交互。云计算是分布式计算（Distributed Computing）、并行计算（Parallel Computing）、效用计算（Utility Computing）、网络存储（Network Storage Technologies）、虚拟化（Virtualization）、负载均衡（Load Balance）、热备份冗余（High Available）等传统计算机和网络技术发展融合的产物。

在传统模式下，企业建立一套 IT 系统不仅需要购买硬件等基础设施，还要购买软件的许可证，并且需要专门的人员维护。当企业的规模扩大时还要继续升级各种软硬件设施以满足需要。对于企业来说，计算机的硬件和软件本身并非他们真正需要的，它们仅仅是完成工作、提供效率的工具而已。对个人来说，正常使用计算机需要安装许多软件，而许多软件是收费的，对不经常使用该软件的用户来说购买是非常不划算的。可不可以有这样的服务，能够提供需要的所有软件供用户租用？这样，用户只需要在用的时候付少量“租金”即可“租用”到这些软件服务，从而节省许多购买软件和硬件的资金。

云计算的最终目标是将计算、服务和应用作为一种公共设施提供给公众，使人们能够像使用水、电、煤气和电话那样使用计算机资源。

云计算模式即为电厂集中供电模式。在云计算模式下，用户的计算机会变得十分简单，或许不大的内存、不需要硬盘和各种应用软件，就可以满足用户的需求，因为用户的计算机除了通过浏览器给“云”发送指令和接收数据外基本上什么都不用做便可以使用云服务提供商的计算资源、存储空间和各种应用软件。这就像连接“显示器”和“主机”的电缆无限长，从而可以把显示器放在使用者的面前，而主机放在远到甚至计算机使用者本人也不知道的地方。云计算把连接“显示器”和“主机”的电缆变成了网络，把“主机”变成云服务提供商的服务器集群。

在云计算环境下，用户的使用观念也会发生彻底的变化：从“购买产品”到“购买服务”转变，因为他们直接面对的将不再是复杂的硬件和软件，而是最终的服务。用户不需要拥有看得见、摸得着的硬件设施，也不需要为机房支付设备供电、空调制冷、专人维护等费用，并且不需要等待漫长的供货周期、项目实施等冗

长的时间，只需要把钱汇给云计算服务提供商，用户将会马上得到需要的服务。

2. 云计算服务

云计算服务，即云服务。中国云计算服务网的定义是：可以拿来作为服务提供使用的云计算产品，包括云主机、云空间、云开发、云测试和综合类产品等。如果将云计算形容为水、电、煤这些基础资源，那么云服务就可以形容为利用这些基础资源而实施的服务。大多数用户并不关心这些基础资源是怎么来的，而是关心可以用来做什么。从这个角度来看，云服务比云计算更加大众，也意味着更大的市场。云服务的诞生前提是互联网打破地域分割，形成一个统一大市场，为个性化需求提供产品开始有利可图。云服务的普及可以达到的客观效果是：把创业成本降到最低，创业者只专注于创意等核心环节，运营和管理将不再重要。小公司开始挑战大公司，颠覆“规模制胜”的工业文明，从而推动社会和文化将更加独立和自由。

云计算服务可以认为包括基础设施即服务（IaaS）、平台即服务（PaaS）和软件即服务（SaaS）多个层次，如图 1-1-1 所示。

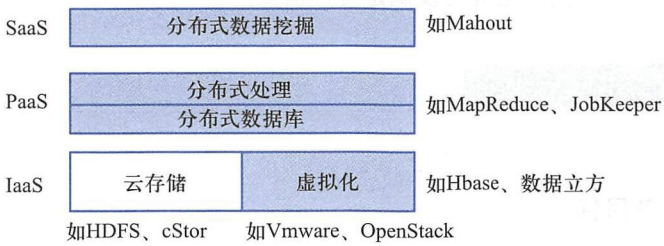


图 1-1-1　云计算服务层次结构图

（1）IaaS

IaaS（Infrastructure-as-a-Service）：基础设施即服务，消费者通过 Internet 可以从完善的计算机基础设施获得服务。IaaS 是把数据中心、基础设施等硬件资源通过 Web 分配给用户的商业模式。

（2）PaaS

PaaS（Platform-as-a-Service）：平台即服务。可以提供软件开发所需的基础功能模块，特别是非核心模块，但又有普遍需求的模块，例如通信、存储、推送等。PaaS 的公司在网上提供各种开发和分发应用的解决方案，按需使用云端的功能模块既能够免去繁琐的开发维护工作，又能提升客户体验，帮助企业专注于自己核心业务。但是相对于发展时间较长的 IaaS 和 SaaS 来说，国内的 PaaS 发展程度相对比较低。

（3）SaaS

SaaS（Software-as-a-Service）：软件即服务。它是一种通过 Internet 提供软件的模式，用户无需购买软件，而是向提供商租用基于 Web 的软件来管理企业经营活动。

国内比较知名的云服务厂商有腾讯云、阿里云、百度云、华为云、盛大云等。

3. 云应用

“云应用”是“云计算”概念的子集，是云计算技术在应用层的体现。云应用跟云计算最大的不同在于，云计算作为一种宏观技术发展概念而存在，而云应用则

是直接面对客户解决实际问题的产品。“云应用”的工作原理是把传统软件“本地安装、本地运算”的使用方式变为“即取即用”的服务，通过互联网或局域网连接并操控远程服务器集群，完成业务逻辑或运算任务的一种新型应用。“云应用”的主要载体为互联网技术，以瘦客户端（Thin Client）或智能客户端（Smart Client）的展现形式，其界面实质上是 HTML5、JavaScript 或 Flash 等技术的集成。云应用不但可以帮助用户降低 IT 成本，更能大大提高工作效率，因此传统软件向云应用转型的发展革新浪潮已经不可阻挡。

云应用可以看作是 SaaS 的升级。相比 SaaS，云应用的发展拥有天时。从宏观行业发展趋势看，云计算已经被国家列为“十二五”规划的重点支持领域，媒体的报道也铺天盖地，这必然会形成巨大的市场爆发力。同时，云应用的发展也拥有地利。随着科学技术的进步，优秀的云应用可以媲美传统软件的强大功能。一些在个人市场取得成功的互联网公司也开始出现在企业云应用这个新兴市场。最后，云应用的发展还拥有人和。云应用不会仅局限在公有云上，针对一些数据较为敏感的企业，私有云应用可以更好地迎合及满足客户需求。

任务 1.2　认识 Docker

微课 1-2
认识 Docker

【任务目标】

- 了解 Docker 的起源与发展。
- 了解 Docker 的组成架构。
- 了解 Docker 与虚拟机的区别。

【技术要点】

1. Docker 的起源与发展

Docker 的前身是一家 dotCloud 的小公司，主要业务是为开发者或开发商提供基于 PaaS 平台的技术服务，并提供开发工具和技术框架。

2011 年该公司拿到了 1000 万美元的融资，然而和谷歌、亚马逊等大公司相比还是杯水车薪。

dotCloud 公司的创始人 Solomon Hykes 做了一个决定：将 dotCloud 的核心引擎开源，并在同年 3 月就发布了 Docker 0.1 版本，从决定开源到发布第一个版本只用了一个月的时间。从此之后，几乎每个月都会发布一个新的 Docker 版本。

2017 年 3 月 3 日，Docker 官方发表了一篇博客，Docker 版本从 1.13 直接跳入 17.03，该版本的意思是 2017 年 3 月。同时，还声明了 Docker 以后会以 CE（Community Edition）和 EE（Enterprise Edition）的形式发布。

在用心的经营下，Docker 的技术讨论社区十分活跃，Docker 团队从社区获得了很多好的建议，可以说，Docker 的成功起于开源，发于社区。

2. Docker 的组成架构

Docker 是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包

到一个可移植的容器中，然后发布到任何流行的 Linux 机器上。Docker 是一个重新定义了程序开发测试、交付和部署过程的开放平台，Docker 可以称为构建一次，到处运行，这就是 Docker 提出的“Build once，Run anywhere”。

Docker 由以下部分组成。

（1）Docker Client 客户端和 Docker daemon 守护进程

Docker 是 C/S（客户端 client/服务器 server）架构模式。Docker 通过客户端连接守护进程，通过命令向守护进程发出请求，守护进程通过一系列的操作返回结果。Docker 客户端可以连接本地或者远程的守护进程。Docker 客户端和服务器通过 socket 或 RESTful API 进行通信，如图 1-2-1 所示。

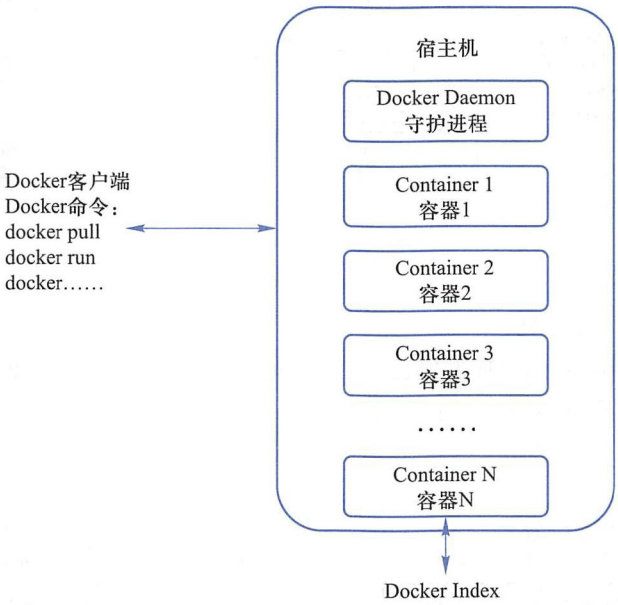


图 1-2-1　客户端和守护进程关系图

（2）Docker Image 镜像

文件的层次结构，以及包含如何运行容器的元数据，Dockerfile 中的每条命令都会在文件系统中创建一个新的层次结构，文件系统在这些层次上构建起来，镜像就构建在这些联合的文件系统之上。Docker 镜像就是一个只读的模板，镜像可以用来创建 Docker 容器。Docker 提供了一个很简单的机制来创建镜像或者更新现有的镜像，用户甚至可以直接从其他人那里下载一个已经做好的镜像来直接使用。

（3）Docker Container 容器

容器是从镜像创建的运行实例。它可以被启动、开始、停止、删除。每个容器都是相互隔离的、保证安全的平台。可以把容器看做是一个简易版的 Linux 环境，Docker 利用容器来运行应用。

（4）Docker Registry 仓库

仓库是集中存放镜像文件的场所，仓库注册服务器（Registry）上往往存放着多个仓库，每个仓库中又包含了多个镜像，每个镜像有不同的标签（Tag）。目前，最大的公开仓库是 Docker Hub，存放了数量庞大的镜像供用户下载。

Docker 仓库用来保存用户个人的 images，当用户创建了自己的 image 之后就可

以使用 push 命令将它上传到公有或者私有仓库，这样下次要在另外一台机器上使用该 image 时，只需要从仓库上使用 pull 命令即可。

这几部分的关系如图 1-2-2 所示。

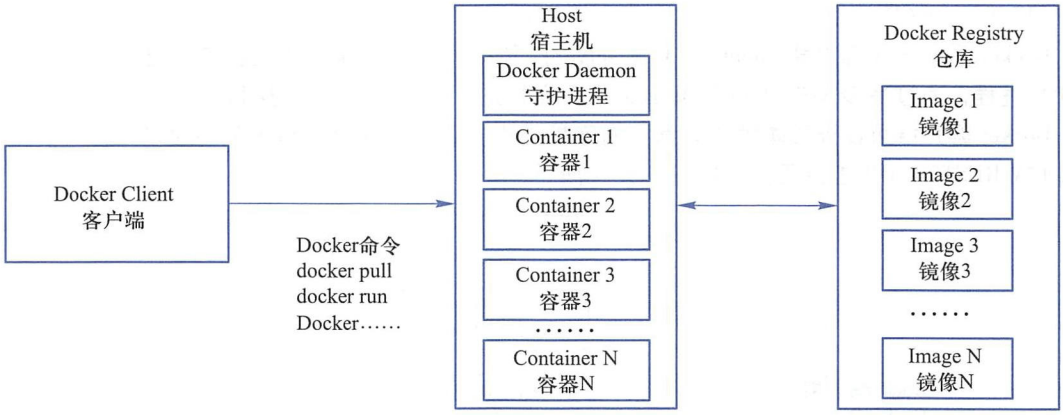


图 1-2-2　Docker 组成架构

3. Docker 与虚拟机

虚拟机和 Docker 的实现框架如图 1-2-3 所示，图 1-2-3a 是虚拟机的实现框架，图 1-2-3b 是 Docker 引擎的实现框架。

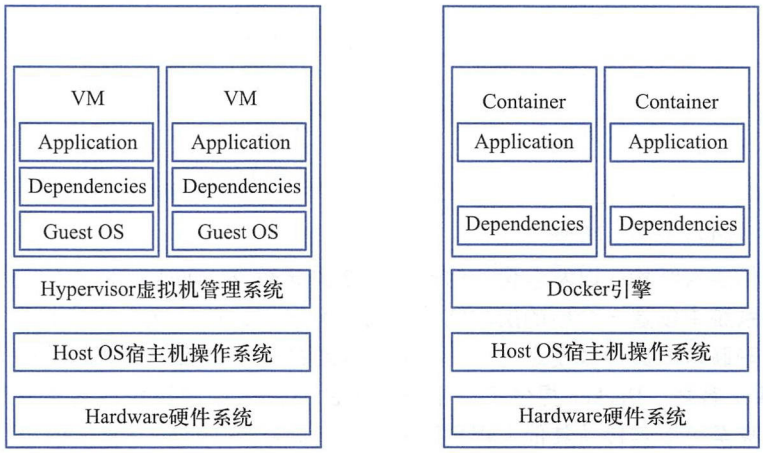


图 1-2-3　虚拟机和 Docker 实现框架结构图

对比虚拟机与 Docker，Docker 守护进程可以直接与主操作系统进行通信，为各个 Docker 容器分配资源；它还可以将容器与主操作系统隔离，并将各个容器互相隔离。虚拟机启动需要数分钟，而 Docker 容器可以在数毫秒内启动。由于没有臃肿的客户机操作系统，Docker 可以节省大量的磁盘空间以及其他系统资源。

表 1-2-1 详细列举出了 Docker 与虚拟机各项性能指标的对比。

表 1-2-1　Docker 与虚拟机各项性能指标的对比

	Docker 容器	虚拟机（VM）
操作系统	与宿主机共享 OS	宿主机 OS 上运行虚拟机 OS
存储大小	镜像小，便于存储与传输	镜像庞大（vmdk、vdi 等）
运行性能	几乎无额外性能损失	操作系统额外的 CPU、内存消耗
可移植性	轻便灵活，适应于 Linux	笨重，与虚拟化技术耦合度高
硬件亲和性	面向软件开发者	面向硬件运维者
部署速度	快速，秒级	较慢，平均 10 秒以上

说明：

虽然 Docker 具有自身的优势，但也不能完全否定虚拟机技术，因为两者有不同的使用场景。虚拟机更擅长于彻底隔离整个运行环境。例如，云服务提供商通常采用虚拟机技术隔离不同的用户。而 Docker 通常用于隔离不同的应用，如前端、后端以及数据库。

Docker 有以下几种应用场景。

- 面向开发人员：快速开发、交付应用程序。开发环境的机器内存通常比较小，之前使用虚拟的时候，经常需要为开发环境的机器加内存，而现在 Docker 可以轻易地让几十个服务在 Docker 中运行起来。
- 面向运维人员：降低运维成本。正如通过虚拟机来整合多个应用，Docker 隔离应用的能力使得 Docker 可以整合多个服务器以降低成本。Docker 通过镜像机制，将代码和运行环境直接打包成镜像，放到容器启动即可。
- 面向企业：Docker 本身就起源于 PaaS，在 Docker 应用中，面向企业是可以提供 Paas 层的实现。

调查显示，目前企业对 Docker 的接受程度在不断提高。但 Docker 的安全性似乎仍旧是企业顾虑的主要原因。随着容器技术逐渐得到 IT 界的认可，CaaS（Container as a Service，容器即服务）也逐渐形成。Docker 作为一项年轻的技术，很多专家认为其安全性可能还不够成熟，总体上 Docker 的安全性能还不错，只是这还是一项年轻的技术，因此目前尚未积累起能够满足实际生产需求的完整工具生态系统。

项目总结

在本项目中介绍了云计算、云应用以及 Docker 等相关概念，学习者通过本项目的学习了解到云计算、云应用，Docker 引擎等相关技术概念。学习者通过本项目的学习对自己所接触的行业、专业以及未来技术发展方向有一个顶层的认知和宏观的理解。

项目 2　开发技术概述

PPT 开发技术概述

项目描述

本项目详细讲解了系统开发过程中所用到的各项框架技术，包括前台网页框架 Bootstrap、Spring 框架、SpringMVC 框架、MyBatis 框架、jQuery 与 Ajax 技术以及 E-Charts 插件。

知识目标

- 了解 Bootstrap 网页框架结构及用法。
- 了解 Spring 框架作用及原理。
- 了解 SpringMVC 框架作用及原理。
- 了解 MyBatis 框架作用及原理。
- 了解 jQuery 与 Ajax 异步响应技术。
- 了解 E-Charts 图标显示插件。

任务列表

任务编号	任务名称	建议课时
任务 2.1	认识 Bootstrap 网页框架	
任务 2.2	认识 Ajax 技术	
任务 2.3	认识 Spring 框架	
任务 2.4	认识 SpringMVC 框架	
任务 2.5	认识 MyBatis 框架	
任务 2.6	认识 ECharts 图表插件	
	总计：	4 课时

任务 2.1　认识 Bootstrap 网页框架

微课 2-1
认识 Bootstrap 网页框架

【任务目标】

- 了解 Bootstrap 网页框架。
- 了解 Bootstrap 网页框架网格结构。
- 了解 Bootstrap 表格、表单、按钮。

【技术要点】

1. Bootstrap 简介

Bootstrap 是一个用于快速开发 Web 应用程序和网站的前端框架。Bootstrap 提供了 Web 界面开发的基本模块，包括 Grid（网格）、Typography（排版）、Tables（表单）、Forms（样式）、Buttons（按钮）和 Responsiveness（按钮和响应式工具）、Dropdowns（下拉菜单）、Navigation（导航栏）、Modals（模态框）、Pagination（分页）、Carousal（轮播）、Breadcrumb（面包屑导航）、Tab（标签页）、Headers（标题）等组件。

通过这些组件，开发人员无需掌握太多的 HTML 和 CSS（Cascading Style Sheets，层叠样式表）知识，就可以快速地搭建起来一个 Web 界面。需要指出地是，Bootstrap 是一个通用的框架，就像任何其他通用的东西，如果需要对该框架进行定制开发，就需要开发人员具备良好的 HTML 和 CSS 基础，才能对这个框架进行深入研究。

2. Bootstrap 的网格结构

Bootstrap 提供了一套响应式、移动设备优先的流式网格系统，随着屏幕的拉伸，系统会自动分为最多 12 列，如图 2-1-1 所示。

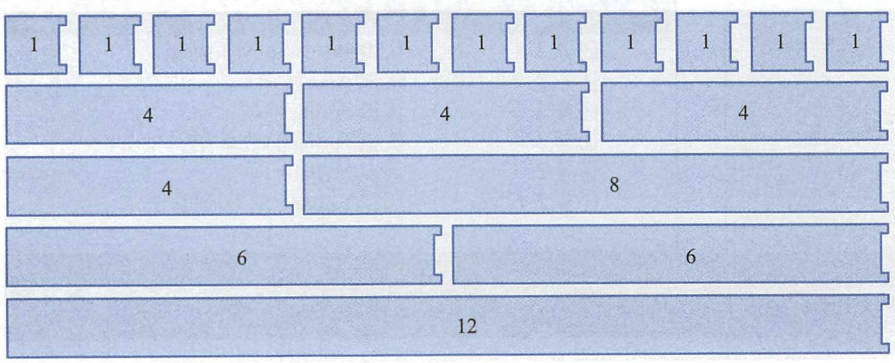


图 2-1-1　Bootstrap 的流式网格系统

(1) 网格选项

表 2-1-1 总结了 Bootstrap 网格系统是如何跨多个设备工作的。

表 2-1-1　Bootstrap 网格系统

	超小设备手机	小型设备平板电脑	中型设备台式电脑	大型设备台式电脑
网格行为	一直是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的	以折叠开始，断点以上是水平的
最大容器宽度	None（auto）	750px	970px	1170px
Class 前缀	.col-xs-	.col-sm-	.col-md-	.col-lg-
列数量和	12	12	12	12
最大列宽	Auto	60px	78px	95px
间隙宽度	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）	30px（一个列的每边分别 15px）
可嵌套	Yes	Yes	Yes	Yes
偏移量	Yes	Yes	Yes	Yes
列排序	Yes	Yes	Yes	Yes

（2）基本的网格结构

以下是 Bootstrap 网格的基本结构。

```
<div class="container">
<div class="row">
<div class="col-*-*"></div>
<div class="col-*-*"></div>
</div>
<div class="row">...</div>
</div>
```

3. Bootstrap 的表格

Bootstrap 提供了一个清晰的创建表格的布局。

（1）表格类

Bootstrap 的表格样式见表 2-1-2。

表 2-1-2　表 格 样 式

类	描　　述
.table	为任意<table>添加基本样式（只有横向分隔线）
.table-striped	在<tbody>内添加斑马线形式的条纹（IE8 不支持）
.table-bordered	为所有表格的单元格添加边框
.table-hover	在<tbody>内的任一行启用鼠标悬停状态
.table-condensed	让表格更加紧凑

（2）<tr>、<th>和<td>类

Bootstrap 表格的提式样式见表 2-1-3。

表 2-1-3　Bootstrap 表格的提示样式

类	描　述
. active	将悬停的颜色应用在行或者单元格上
. success	表示成功的操作
. info	表示信息变化的操作
. warning	表示一个警告的操作
. danger	表示一个危险的操作

（3）基本的表格

如果想要实现一个只带有内边距（padding）和水平分割的基本表，可添加 class. table，如下面实例代码所示。

```
<table class = " table" >
<caption>基本的表格布局</caption>
<thead>
    <tr><th>名称</th><th>城市</th></tr>
</thead>
<tbody>
    <tr><td>Tanmay</td><td>Bangalore</td></tr>
    <tr><td>Sachin</td><td>Mumbai</td></tr>
</tbody>
</table>
```

运行结果如图 2-1-2 所示。

基本的表格布局

名称	城市
Tanmay	Bangalore
Sachin	Mumbai

图 2-1-2　表格布局效果图

4. Bootstrap 的表单

（1）垂直表单

基本的表单结构是 Bootstrap 自带的，个别的表单控件自动接收一些全局样式。下面列出了创建基本表单的步骤。

① 向父<form>元素添加 role = " form" 。

② 把标签和控件放在一个带有 class . form-group 的 中。这是获取最佳间距所必需的。

③ 向所有的文本元素<input>、<textarea>和<select>添加 class = " form-control" 。

垂直表单如下面实例代码所示。

```
<form role = " form" >
<div class = " form-group" >
```

```
<label for="name">名称</label>
<input type="text" class="form-control" id="name" placeholder="请输入名称">
</div>
<div class="form-group">
<label for="inputfile">文件输入</label>
<input type="file" id="inputfile">
<p class="help-block">这里是块级帮助文本的实例。</p>
</div>
<div class="checkbox">
<label><input type="checkbox">请打勾</label>
</div>
<button type="submit" class="btn btn-default">提交</button>
</form>
```

运行结果如图 2-1-3 所示。

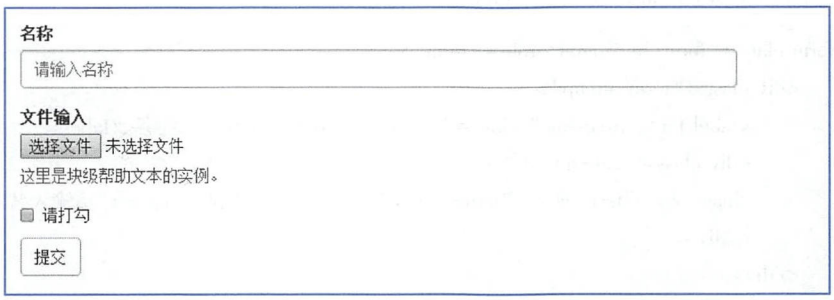


图 2-1-3　垂直表单效果图

(2) 内联表单

如果需要创建一个表单，它的所有元素是内联的，向左对齐的，标签是并排的，向<form>标签添加 class . form-inline，代码如下。

```
<form class="form-inline" role="form">
    <div class="form-group">
        <label class="sr-only" for="name">名称</label>
        <input type="text" class="form-control" id="name" placeholder="请输入名称">
    </div>
    <div class="form-group">
        <label class="sr-only" for="inputfile">文件输入</label>
        <input type="file" id="inputfile">
    </div>
    <div class="checkbox">
        <label><input type="checkbox">请打勾</label>
    </div>
    <button type="submit" class="btn btn-default">提交</button>
</form>
```

运行结果如图 2-1-4 所示。

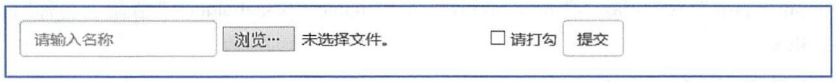


图 2-1-4　内联表单效果图

• 默认情况下，Bootstrap 中的 input、select 和 textarea 有 100% 宽度。在使用内联表单时，需要在表单控件上设置一个宽度。

• 使用 class . sr-only，可以隐藏内联表单的标签。

(3) 水平表单

水平表单与其他表单不仅标记的数量上不同，而且表单的呈现形式也不同。例如，创建一个水平布局的表单，按以下几个步骤进行。

① 向父<form>元素添加 class . form-horizontal。

② 把标签和控件放在一个带有 class . form-group 的 中。

③ 向标签添加 class . control-label。

```
<form class="form-horizontal" role="form">
    <div class="form-group">
        <label for="firstname" class="col-sm-2 control-label">名字</label>
        <div class="col-sm-10">
        <input type="text" class="form-control" id="firstname" placeholder="请输入名字">
        </div>
    </div>
    <div class="form-group">
        <label for="lastname" class="col-sm-2 control-label">姓</label>
        <div class="col-sm-10">
        <input type="text" class="form-control" id="lastname" placeholder="请输入姓">
        </div>
    </div>
    <div class="form-group">
        <div class="col-sm-offset-2 col-sm-10">
            <div class="checkbox">
                <label><input type="checkbox">请记住我</label>
            </div>
        </div>
    </div>
    <div class="form-group">
        <div class="col-sm-offset-2 col-sm-10">
            <button type="submit" class="btn btn-default">登录</button>
        </div>
    </div>
</form>
```

运行结果如图 2-1-5 所示。

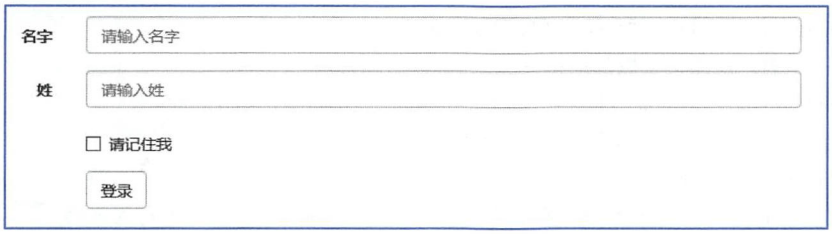


图 2-1-5　水平表单效果图

5. Bootstrap 的按钮

(1) 按钮类

Bootstrap 提供了一些选项来定义按钮的样式，这些样式也可以应用到<a>、<button>和<input>元素上，见表 2-1-4。

表 2-1-4　Bootstrap 的按钮类

类	描　述
.btn	为按钮添加基本样式
.btn-default	默认/标准按钮
.btn-primary	原始按钮样式（未被操作）
.btn-success	表示成功的动作
.btn-info	该样式可用于要弹出信息的按钮
.btn-warning	表示需要谨慎操作的按钮
.btn-danger	表示一个危险动作的按钮操作
.btn-link	让按钮看起来像个链接（仍然保留按钮行为）
.btn-lg	制作一个大按钮
.btn-sm	制作一个小按钮
.btn-xs	制作一个超小按钮
.btn-block	块级按钮（拉伸至父元素 100%的宽度）
.active	按钮被点击
.disabled	禁用按钮

下面的实例演示了上面所有的按钮 class：

```
<! --标准的按钮 -->
<button type="button" class="btn btn-default">默认按钮</button>
<! --提供额外的视觉效果,标识一组按钮中的原始动作 -->
<button type="button" class="btn btn-primary">原始按钮</button>
<! --表示一个成功的或积极的动作 -->
<button type="button" class="btn btn-success">成功按钮</button>
<! --信息警告消息的上下文按钮 -->
```

```
<button type="button" class="btn btn-info">信息按钮</button>
<!--表示应谨慎采取的动作 -->
<button type="button" class="btn btn-warning">警告按钮</button>
<!--表示一个危险的或潜在的负面动作 -->
<button type="button" class="btn btn-danger">危险按钮</button>
<!--并不强调是一个按钮,看起来像一个链接,但同时保持按钮的行为 -->
<button type="button" class="btn btn-link">链接按钮</button>
```

运行结果如图 2-1-6 所示。

图 2-1-6　Bootstrap 按钮样式

(2) 按钮组

按钮组允许多个按钮被堆叠在同一行上。当需要把按钮对齐在一起时，这就显得非常有用。可以通过 Bootstrap 按钮（Button）插件添加可选的 JavaScript 单选框和复选框样式行为。表 2-1-5 列出了 Bootstrap 提供的使用按钮组的一些重要 class。

表 2-1-5　Bootstrap 按钮组类

Class	描　述
.btn-group	该 class 用于形成基本的按钮组。在 .btn-group 中放置一系列带有 class. btn 的按钮
.btn-toolbar	该 class 有助于把几组 结合到一个<divclass="btn-toolbar">中，获得更复杂的组件
.btn-group-lg, .btn-group-sm, .btn-group-xs	这些 class 可应用到整个按钮组的大小调整，而不需要对每个按钮进行大小调整
.btn-group-vertical	该 class 可以使一组按钮垂直堆叠显示

(3) 基本的按钮组

下面的实例演示了上面表格中讨论到的 class. btn-group 的使用：

```
<div class="btn-group">
    <button type="button" class="btn btn-default">按钮 1</button>
    <button type="button" class="btn btn-default">按钮 2</button>
    <button type="button" class="btn btn-default">按钮 3</button>
</div>
```

运行结果如图 2-1-7 所示。

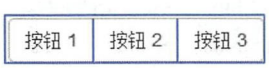


图 2-1-7　Bootstrap 按钮组

任务 2.2　认识 Ajax 技术

【任务目标】

- 了解 Ajax 异步通信技术。
- 了解通过 jQuery 实现 Ajax。

微课 2-2
认识 Ajax 技术

【技术要点】

1. Ajax 简介

Ajax（Asynchronous Javascript And XML，异步 JavaScript 和 XML），是指一种创建交互式网页应用的网页开发技术，它不是新的编程语言，而是一种使用现有标准的新方法。Ajax 是在不重新加载整个页面的情况下，与服务器交换数据并更新部分网页的技术。涉及到的技术包括 HTML、CSS、JavaScript、jQuery。

该技术在 1998 年前后得到了应用。允许客户端脚本发送 HTTP 请求（XMLHTTP）的第一个组件由 Outlook Web Access 小组写成。该组件原属于微软 Exchange Server，并且迅速地成为了 Internet Explorer 4.0 的一部分。在 2005 年初，许多事件使得 Ajax 被大众所接受。谷歌在其著名的交互应用程序中使用了异步通信，如 Google、Google 地图、Google 搜索、Gmail 等。

Ajax 前景非常乐观，可以提高系统性能，优化用户界面。Ajax 现有直接框架 AjaxPro，可以引入 AjaxPro.2.dll 文件，拥有可以直接在前台页面 JS 调用后台页面的方法。但此框架与 FORM 验证有冲突。另外微软也引入了 Ajax 组件，其需要添加 AjaxControlToolkit.dll 文件，才可以在控件列表中出现相关控件。

2. jQuery 中的$ajax()方法

jQuery 是一个快速、简洁的 JavaScript 框架，是继 Prototype 之后又一个优秀的 JavaScript 框架。jQuery 设计的宗旨是“Write Less，Do More”，倡导写更少的代码，做更多的事情。它封装 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化 HTML 文档操作、事件处理、动画设计和 Ajax 交互。jQuery 的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的 CSS 选择器，并且可对 CSS 选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。

ajax()方法通过 HTTP 请求加载远程数据。该方法是 jQuery 底层 Ajax 实现。$.ajax()返回其创建的 XMLHttpRequest 对象。语法格式如下：

```
jQuery.ajax([settings])
```

settings：可选。用于配置 Ajax 请求的键值对集合，其参数明细表详解见表 2-2-1。

表 2-2-1　$ajax()参数明细表

参　数	描　述
options	类型：Object 可选。Ajax 请求设置。所有选项都是可选的

续表

参　数	描　述
async	类型：Boolean 默认值：true。默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。 注意：同步请求将锁住浏览器，用户的其他操作必须等待请求完成才可以执行
beforeSend（XHR）	类型：Function 发送请求前可修改 XMLHttpRequest 对象的函数，如添加自定义 HTTP 头。XMLHttpRequest 对象是唯一的参数。如果返回 false 可以取消本次 Ajax 请求
cache	类型：Boolean 默认值：true。dataType 为 script 和 jsonp 时，默认为 false。设置为 false 将不缓存此页面。 **jQuery 1.2 新功能**
complete（XHR，TS）	类型：Function 请求完成后回调函数（请求成功或失败之后均调用）。 参数：XMLHttpRequest 对象和一个描述请求类型的字符串
contentType	类型：String 默认值："application/x-www-form-urlencoded"。发送信息至服务器时内容编码类型。默认值适合大多数情况。如果明确地传递了一个 content-type 给 $.ajax()，那么它必定会发送给服务器（即使没有数据要发送）
context	类型：Object 该对象用于设置 Ajax 相关回调函数的上下文。即让回调函数内 this 指向这个对象（如果不设定这个参数，那么 this 就指向调用本次 Ajax 请求时传递的 options 参数）。例如指定一个 DOM 元素作为 context 参数，这样就设置了 success 回调函数的上下文为该 DOM 元素
data	类型：String 发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称，如｛foo：["bar1"，"bar2"]｝转换为'&foo=bar1&foo=bar2'
dataFilter	类型：Function 给 Ajax 返回的原始数据的进行预处理的函数。提供 data 和 type 两个参数：data 是 Ajax 返回的原始数据，type 是调用 jQuery.ajax 时提供的 dataType 参数。函数返回的值将由 jQuery 进一步处理
dataType	类型：String 预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息来智能判断。 "xml"：返回 XML 文档，可用 jQuery 处理。 "html"：返回纯文本 HTML 信息。包含的 script 标签会在插入 DOM 时执行。 "script"：返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了"cache"参数。 注意：在远程请求时（不在同一个域下），所有 POST 请求都将转为 GET 请求。（因为将使用 DOM 的 script 标签来加载） "json"：返回 JSON 数据。 "jsonp"：JSONP 格式。使用 JSONP 形式调用函数时，如"myurl?callback=?" jQuery 将自动替换“?”为正确的函数名，以执行回调函数。 "text"：返回纯文本字符串

续表

参　数	描　述
error	类型：Function 默认值：自动判断（XML 或 HTML）。请求失败时调用此函数。 有 XMLHttpRequest 对象、错误信息、（可选）捕获的异常对象 3 个参数。 如果发生了错误，错误信息（第 2 个参数）除了得到 null 之外，还可能是"timeout"、"error"、"notmodified" 和"parsererror"
global	类型：Boolean 是否触发全局 Ajax 事件。默认值：true。设置为 false 将不会触发全局 Ajax 事件，如 ajaxStart 或 ajaxStop 可用于控制不同的 Ajax 事件
ifModified	类型：Boolean 仅在服务器数据改变时获取新数据。默认值：false。使用 HTTP 包 Last-Modified 头信息判断。在 jQuery 1.4 版本中，它也会检查服务器指定的'etag'来确定数据没有被修改过
jsonp	类型：String 在一个 jsonp 请求中重写回调函数的名字。这个值用来替代在 "callback=?" 这种 GET 或 POST 请求中 URL 参数里的 "callback" 部分，如{jsonp:'onJsonPLoad'} 会导致将 "onJsonPLoad=?" 传给服务器
jsonpCallback	类型：String 为 jsonp 请求指定一个回调函数名。这个值将用来取代 jQuery 自动生成的随机函数名。这主要用来让 jQuery 生成独特的函数名，这样管理请求更容易，也能方便地提供回调函数和错误处理。开发者也可以在想让浏览器缓存 GET 请求的时候，指定这个回调函数名
password	类型：String 用于响应 HTTP 访问认证请求的密码
processData	类型：Boolean 默认值：true。默认情况下，通过 data 选项传递进来的数据，如果是一个对象（技术上讲只要不是字符串），都会处理转化成一个查询字符串，以配合默认内容类型"application/x-www-form-urlencoded"。如果要发送 DOM 树信息或其他不希望转换的信息时设置为 false
success	类型：Function 请求成功后的回调函数。 参数：由服务器返回，并根据 dataType 参数进行处理后的数据；描述状态的字符串。 注意：这是一个 Ajax 事件
type	类型：String 默认值："GET"。请求方式为"POST" 或"GET"。 注意：其他 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持
url	类型：String 默认值：当前页地址。发送请求的地址
username	类型：String 用于响应 HTTP 访问认证请求的用户名

一个简单的 jQuery 中$ajax( )方法的实例。前台 JS 代码如下。

```
var url = "/test/check";
$. ajax( {
type: "post",
url: url,
data: {"para":1},
cache: false,
async : false,
dataType: "json",
success: function (data ,textStatus, jqXHR)
{
    if("true" = =data. flag){
        alert("合法!");
        return true;
    }else{
        alert("不合法! 错误信息如下:"+data. errorMsg);
        return false;
    }
  },
  error:function (XMLHttpRequest, textStatus, errorThrown) {
        alert("请求失败!");
  }
});
```

后台 Java 代码如下。

```
public void check( ){
Map<String,String> result = new HashMap<String,String>( );
boolean flag = false;
try{
    String para= getPara("para");
    //校验代码...
    result. put("flag", "true");
    renderJson(result);//返回 json 数据
}catch(Exception e){
  result. put("flag", flag+"");
  result. put("errorMsg", e. getMessage( ));
  renderJson(result);//返回 json 数据
  e. printStackTrace( );
}
}
```

常用的$. get 和$. post 方法是 jQuery 简单易用的高层实现，使用$. get $. post 方法时，jQuery 会自动封装调用底层的$. ajax。$. get 只处理简单的 GET 请求功能以取代复杂$. ajax，请求成功时可调用回调函数。不支持出错时执行函数，否则必须使

用$. ajax。$. post 只处理 post 请求功能以取代复杂$. ajax。请求成功时可调用回调函数。不支持出错时执行函数，否则必须使用$. ajax。

特别注意：dataType 参数的两种值：json 和 jsonp。

json 是理想的数据交换格式，但无法跨域直接获取，于是就将 json 包裹（padding）在一个合法的 js 语句中作为 js 文件传过去。jsonp 全称为 json with padding，很形象地描述出把 json 对象用符合 js 语法的形式包裹起来以使其他网站可以请求得到的过程，也就是将 json 数据封装成 js 文件。这就是 json 和 jsonp 的区别，json 是想要的东西，jsonp 是达到这个目的而普遍采用的一种方法，当然最终获得和处理的还是 json。所以说 json 是目的，jsonp 只是手段。json 总会用到，而 jsonp 只有在跨域获取数据才会用到。在后续的项目实例中会详细展示 jsonp 的用法。

任务 2.3　认识 Spring 框架

【任务目标】

- 了解 Spring 框架的作用。
- 了解 Spring 框架的基本组成。

【技术要点】

1. Spring 框架简介

Spring 是一个开源框架，它由 Rod Johnson 创建。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB（Enterprise JavaBean，Sun 的 JavaEE 服务器端组件模型）完成的事情。然而，Spring 的用途不仅局限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

Spring 是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

- 轻量——大小与开销方面 Spring 都是轻量的。完整的 Spring 框架可以在一个大小只有 1 MB 多的 JAR 文件里发布。并且 Spring 所需的处理开销也是微不足道的。此外，Spring 是非侵入式的：Spring 应用中的对象不依赖于 Spring 的特定类。
- 控制反转——Spring 通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了 IoC，一个对象依赖的其他对象会通过被动的方式传递进来，而不是该对象自己创建或者查找依赖对象。读者可以认为 IoC 与 JNDI（Java Naming and Directory Interface，Java 命名和目录接口）相反。它不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。
- 面向切面——Spring 提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（如审计 auditing 和事务 transaction 管理）进行内聚性的开发。应用对象只实现它们应该做的（完成业务逻辑），仅此而已。它们并不负责其他的系统级关注点，如日志或事务支持。
- 容器——Spring 包含并管理应用对象的配置和生命周期，在这个意义上它是

一种容器，开发者可以配置每个 bean 如何被创建——基于一个可配置原型（prototype），这个 bean 可以创建一个单独的实例或者每次需要时都生成一个新的实例，以及它们是如何相互关联的。然而，Spring 不应该被混同于传统的重量级的 EJB 容器，它们经常是庞大与笨重的，难以使用。

- 框架——Spring 可以将简单的组件配置、组合成为复杂的应用。在 Spring 中，应用对象被声明式地组合，典型地是在一个 XML 文件里。Spring 也提供了很多基础功能（事务管理、持久化框架集成等），将应用逻辑的开发留给开发人员。

所有 Spring 的这些特征使开发者能够编写更干净、更可管理、更易于测试的代码。它们也为 Spring 中的各种模块提供了基础支持。

2. Spring 框架核心模块

Spring 框架由 7 个模块组成，包括 Spring 核心容器模块、应用上下文模块、AOP（Aspect Oriented Programming，面向切面编程）模块、JDBC（Java DataBase Connectivity，Java 数据库连接）抽象和 DAO（Data Access Object，数据库访问对象）模块、对象/关系映射集成模块、Spring 的 Web 模块、Spring 的 MVC 框架模块。如果作为一个整体，这些模块提供了开发企业应用所需的一切。但不是必须将应用完全基于 Spring 框架。开发人员可以自由地挑选合适的应用的模块而忽略其余的模块。

- 核心容器模块：Spring 框架最基础的部分，它提供了依赖注入（Dependency-Injection）特征来实现容器对 Bean 的管理。这里最基本的概念是 BeanFactory，它是任何 Spring 应用的核心。BeanFactory 是工厂模式的一个实现，它使用 IoC 将应用配置和依赖说明从实际的应用代码中分离出来。

- 应用上下文（Context）模块：核心模块的 BeanFactory 使 Spring 成为一个容器，而上下文模块使它成为一个框架。该模块扩展了 BeanFactory 的概念，增加了对国际化（I18N）消息、事件传播以及验证的支持。另外，该模块提供了许多企业服务，如电子邮件、JNDI 访问、EJB 集成、远程以及时序调度（Scheduling）服务。也包括对模版框架（如 Velocity 和 FreeMarker）集成的支持。

- Spring 的 AOP 模块：Spring 在 AOP 模块中提供了对面向切面编程的丰富支持。该模块是在 Spring 应用中实现切面编程的基础。为了确保 Spring 与其他 AOP 框架的互用性，Spring 的 AOP 支持基于 AOP 联盟定义的 API。AOP 联盟是一个开源项目，它的目标是通过定义一组共同的接口和组件来促进 AOP 的使用以及不同的 AOP 实现之间的互用性。通过访问它们的站点，开发者可以找到关于 AOP 联盟的更多内容。Spring 的 AOP 模块也将元数据编程引入了 Spring。使用 Spring 的元数据支持，开发者可以为源代码增加注释，指示 Spring 在何处以及如何应用切面函数。

- JDBC 抽象和 DAO 模块：使用 JDBC 经常导致大量的重复代码，取得连接、创建语句、处理结果集，然后关闭连接。Spring 的 JDBC 和 DAO 模块抽取了这些重复代码，保持了数据库访问代码的干净简洁，并且可以防止因关闭数据库资源失败而引起的问题。该模块还使用了 Spring 的 AOP 模块，为 Spring 应用中的对象提供了事务管理服务。

- 对象/关系映射集成模块：对那些更喜欢使用对象/关系映射工具而不是直接使用 JDBC 的人，Spring 提供了 ORM（Object Relational Mapping，ORM）模块。Spring 并不试图实现它自己的 ORM 解决方案，而是为几种流行的 ORM 框架提供了集成方案，包括 Hibernate（开放源代码的对象关系映射框架）、JDO（Java Data Ob-

ject，它是 Java 对象持久化的新的规范，也是一个用于存取某种数据仓库中的对象的标准化 API）和 iBATIS SQL 映射。Spring 的事务管理支持这些 ORM 框架中的每一个也包括 JDBC。

- Spring 的 Web 模块：Web 上下文模块建立于应用上下文模块之上，提供了一个适合于 Web 应用的上下文。另外，该模块还提供了一些面向服务支持。例如，实现文件上传的 multipart 请求，它也提供了 Spring 和其他 Web 框架的集成，如 Struts、WebWork。
- Spring 的 MVC 框架：Spring 为构建 Web 应用提供了一个功能全面的 MVC 框架。虽然 Spring 可以很容易地与其他 MVC 框架集成，如 Struts，但 Spring 的 MVC 框架使用 IoC 对控制逻辑和业务对象提供了完全的分离。

任务 2.4　认识 SpringMVC 框架

【任务目标】

- 了解 SpringMVC 框架架构。
- 了解 SpringMVC 的优势。

微课 2-3
认识 SSM 框架

【技术要点】

1. SpringMVC 架构

Spring MVC 属于 SpringFrameWork 的后续产品，也是 Spring 框架的重要组成部分，已经融合在 Spring Web Flow 里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用 Spring 进行 Web 开发时，可以选择使用 Spring 的 SpringMVC 框架或集成其他 MVC 开发框架，如 Struts1、Struts2 等。SpringMVC 框架也是一个基于请求驱动的 Web 框架，并且也使用了前端控制器模式来进行设计，再根据请求映射规则分发给相应的页面控制器（动作/处理器）进行处理。首先看一下 SpringMVC 处理请求的流程，如图 2-4-1 所示。

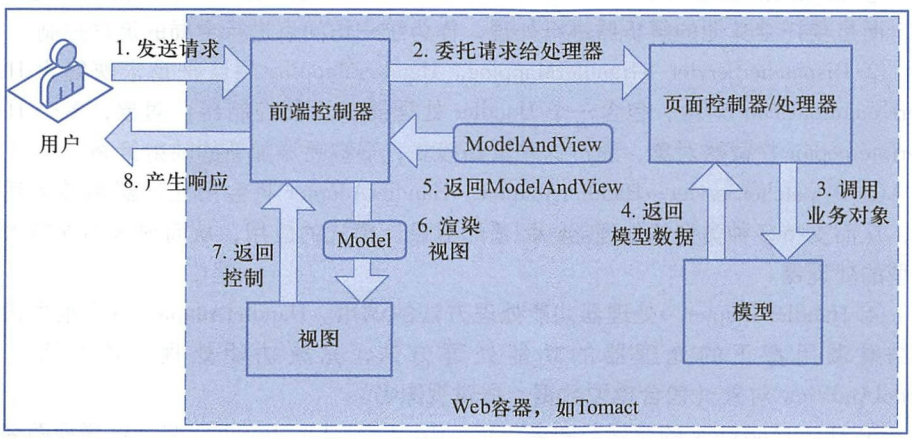


图 2-4-1　SpringMVC 处理请求的流程

（1）具体执行步骤

① 首先用户发送请求发送到前端控制器，前端控制器根据请求信息（如 URL）来决定选择哪一个页面控制器进行处理并把请求委托给它，即以前的控制器的控制逻辑部分，如图 2-4-2 所示的步骤 1 和步骤 2。

② 页面控制器接收到请求后，进行功能处理，首先需要收集和绑定请求参数到一个对象，该对象在 Spring Web MVC 中叫命令对象，并进行验证，然后将命令对象委托给业务对象进行处理。处理完毕后返回一个 ModelAndView（模型数据和逻辑视图名），如图 2-4-2 所示的步骤 3～步骤 5。

③ 前端控制器收回控制权，然后根据返回的逻辑视图名，选择相应的视图进行渲染，并把模型数据传入以便视图渲染，如图 2-4-2 所示的步骤 6 和步骤 7。

④ 前端控制器再次收回控制权，将响应返回给用户，如图 2-4-2 所示的步骤 8。至此整个结束。

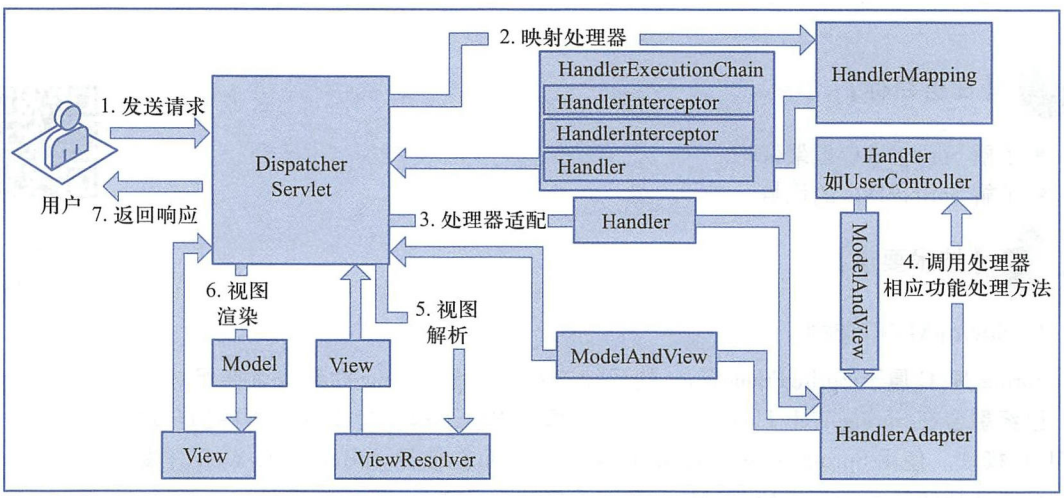


图 2-4-2　Spring Web MVC 核心架构图

（2）核心架构的具体流程步骤

① 首先用户发送请求→DispatcherServlet。前端控制器收到请求后自己不进行处理，而是委托给其他的解析器进行处理，作为统一访问点进行全局的流程控制。

② DispatcherServlet→HandlerMapping。HandlerMapping 将会把请求映射为 HandlerExecutionChain 对象，包含一个 Handler 处理器（页面控制器）对象、多个 HandlerInterceptor 拦截器对象，通过这种策略模式，很容易添加新的映射策略。

③ DispatcherServlet→HandlerAdapter。HandlerAdapter 将会把处理器包装为适配器，从而支持多种类型的处理器，即适配器设计模式的应用，从而很容易支持很多类型的处理器。

④ HandlerAdapter→处理器功能处理方法的调用。HandlerAdapter 将会根据适配的结果调用真正的处理器的功能处理方法，完成功能处理，并返回一个 ModelAndView 对象（包含模型数据、逻辑视图名）。

⑤ ModelAndView 的逻辑视图名→ViewResolver，ViewResolver 将把逻辑视图名解析为具体的 View，通过这种策略模式，很容易更换其他视图技术。

⑥ View→渲染，View 会根据传进来的 Model 模型数据进行渲染，此处的 Model 实际是一个 Map 数据结构，因此很容易支持其他视图技术。

⑦ 返回控制权给 DispatcherServlet，由 DispatcherServlet 返回响应给用户，到此一个流程结束。

此处只是讲了核心流程，没有考虑拦截器、本地解析、文件上传解析等。

2. SpringMVC 优势

- 清晰的角色划分。前端控制器（DispatcherServlet）、请求到处理器映射（HandlerMapping）、处理器适配器（HandlerAdapter）、视图解析器（ViewResolver）、处理器或页面控制器（Controller）、验证器（Validator）、命令对象（Command 请求参数绑定到的对象称为命令对象）、表单对象（Form Object 提供给表单展示和提交到的对象称为表单对象）。
- 分工明确，且扩展点相当灵活。
- 由于命令对象就是一个 POJO，无需继承框架特定 API，可以使用命令对象直接作为业务对象。
- 和 Spring 其他框架无缝集成，是其他 Web 框架所不具备的。
- 可适配。通过 HandlerAdapter 可以支持任意的类作为处理器。
- 可定制性。HandlerMapping、ViewResolver 等能够非常简单地定制。
- 功能强大的数据验证、格式化、绑定机制。
- 利用 Spring 提供的 Mock 对象能够非常简单地进行 Web 层单元测试。
- 本地化、主题的解析的支持，使用户更容易进行国际化和主题的切换。
- 强大的 JSP 标签库，使 JSP 编写更容易。

任务 2.5　认识 MyBatis 框架

【任务目标】

- 了解 MyBatis 运行过程。
- 了解 MyBatis 优缺点。

【技术要点】

1. MyBatis 简介

MyBatis 本是阿帕奇（Apache Web 服务器）的一个开源项目 iBATIS，2010 年该项目由 Apache Software Foundation 迁移到了 Google Code，并且改名为 MyBatis。iBATIS 一词来源于 internet 和 abatis 的组合，是一个基于 Java 的持久层框架。iBATIS 提供的持久层框架包括 SQL Maps 和 Data Access Objects（sDAO）。

2. MyBatis 运行过程

MyBatis 应用程序根据 XML 配置文件创建 SqlSessionFactory，SqlSessionFactory 再根据配置（配置来源于两个地方，一处是配置文件，一处是 Java 代码的注解）获取一个 SqlSession。SqlSession 包含了执行 SQL 所需要的所有方法，可以通过 SqlSession

实例直接运行映射的 SQL 语句，完成对数据的增加、删除、修改、查询和事务提交等，用完之后关闭 SqlSession。MyBatis 整个执行流程，如图 2-5-1 所示。

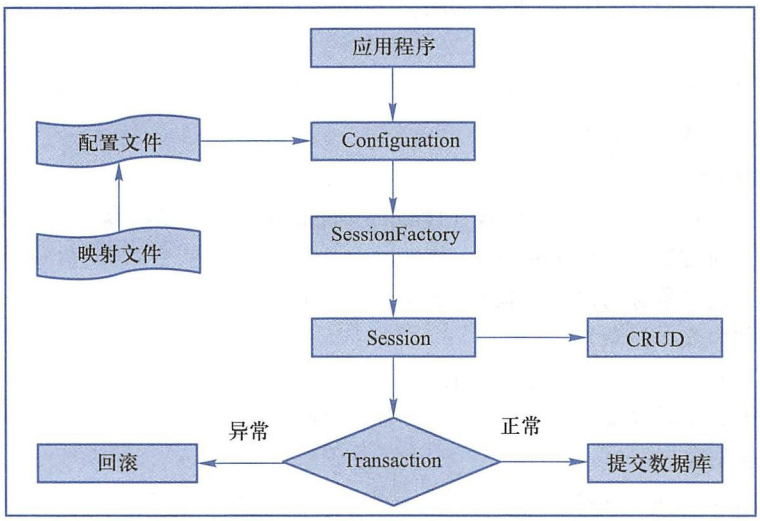


图 2-5-1　MyBatis 执行流程

(1) 加载配置并初始化

触发条件：加载配置文件。

处理过程：将 SQL 的配置信息加载成为一个个 MappedStatement 对象（包括了传入参数映射配置、执行的 SQL 语句、结果映射配置），存储在内存中。

(2) 接收调用请求

触发条件：调用 MyBatis 提供的 API。

传入参数：为 SQL 的 ID 和传入参数对象。

处理过程：将请求传递给下层的请求处理层进行处理。

(3) 处理操作请求

触发条件：API 接口层传递请求过来。

传入参数：为 SQL 的 ID 和传入参数对象。

处理过程：

① 根据 SQL 的 ID 查找对应的 MappedStatement 对象。

② 根据传入参数对象解析 MappedStatement 对象，得到最终要执行的 SQL 和执行传入参数。

③ 获取数据库连接，根据得到的最终 SQL 语句和执行传入参数到数据库执行，并得到执行结果。

④ 根据 MappedStatement 对象中的结果映射配置对得到的执行结果进行转换处理，并得到最终的处理结果。

⑤ 释放连接资源。

(4) 返回处理结果将最终的处理结果返回。

3. MyBatis 的优缺点

(1) 优点

- 简单易学。MyBatis 本身就很小且简单，没有任何第三方依赖，最简单安装只

要两个 jar 文件和配置几个 SQL 映射文件。易于学习和使用，通过文档和源代码，可以掌握其设计思路和实现方法。

- 灵活。MyBatis 不会对应用程序或者数据库的现有设计强加任何影响。SQL 写在 XML 里，便于统一管理和优化。通过 SQL 基本上可以实现用户不使用数据访问框架实现所有的功能。
- 解除 SQL 与程序代码的耦合。通过提供 DAL 层，将业务逻辑和数据访问逻辑分离，使系统的设计更清晰、更易维护、更易单元测试。SQL 和代码的分离，提高了可维护性。
- 提供映射标签。支持对象与数据库的 orm 字段关系映射。
- 提供对象关系映射标签。支持对象关系组建维护。
- 提供 XML 标签。支持编写动态 SQL。

(2) 缺点

- 编写 SQL 语句时工作量很大，尤其是字段多、关联表多时，更是如此。
- SQL 语句依赖于数据库，导致数据库移植性差，不能更换数据库。
- 框架比较简陋，功能尚有缺失，虽然简化了数据绑定代码，但是整个底层数据库查询实际还是需要自己来编写，工作量比较大，而且不太容易适应数据库的快速修改。
- 二级缓存机制不佳。

任务 2.6　认识 ECharts 图表插件

【任务目标】

微课 2-4
认识 ECharts 图表插件

- 了解 ECharts 插件。
- 了解 ECharts 插件的简单使用方法。

【技术要点】

1. ECharts 插件简介

ECharts（Enterprise Charts，商业级数据图表），一个纯 JavaScript 的图表库，可以流畅地运行在 PC 和移动设备上，兼容当前绝大部分浏览器（IE6/7/8/9/10/11、Chrome、Firefox、Safari 等），底层依赖轻量级的 Canvas 类库 ZRender，提供直观、生动、可交互、可高度个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力。支持折线图（区域图）、柱状图（条状图）、散点图（气泡图）、K 线图、饼图（环形图）、雷达图（填充雷达图）、和弦图、力导向布局图、地图、仪表盘、漏斗图、事件河流图 12 类图表，同时提供标题、详情气泡、图例、值域、数据区域、时间轴、工具箱 7 种可交互组件，支持多图表、组件的联动和混搭展现。各种图形效果如图 2-6-1～图 2-6-3 所示。

2. ECharts 简单实例

① 下载 ECharts 插件。从官网下载 ECharts 插件，如图 2-6-4 所示。

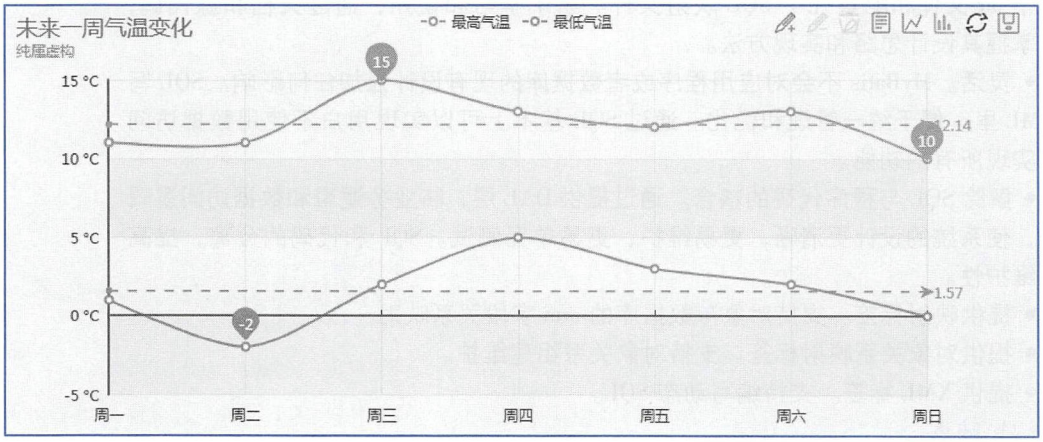


图 2-6-1　ECharts 绘制折线图

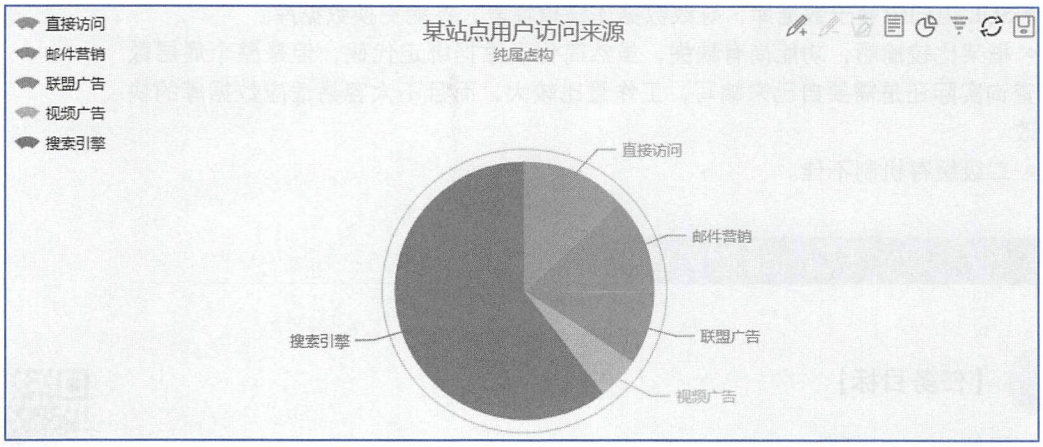


图 2-6-2　ECharts 绘制饼图

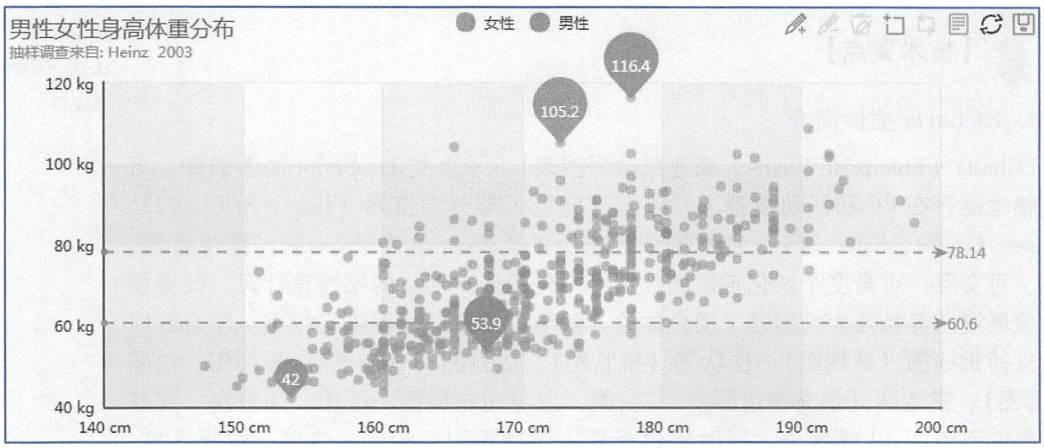


图 2-6-3　ECharts 绘制散点图

② 引入 ECharts。从 ECharts 3 开始不再强制使用 AMD 的方式按需引入，代码里也不再内置 AMD 加载器。因此引入方式简单了很多，只需要像普通的 JavaScript 库一样用 script 标签引入。引入代码如下。

图 2-6-4　ECharts 下载界面

```
<! DOCTYPE html>
<html>
<head>
<meta charset = " utf-8" >
<! --引入 ECharts 文件 -->
<script src = " echarts. min. js" ></script>
</head>
</html>
```

③ 绘制一个简单的图表。在绘图前需要为 ECharts 准备一个具备高宽的 DOM 容器。

```
<body>
<! --为 ECharts 准备一个具备大小(宽高)的 DOM -->
<div id = " main"  style = " width: 600px;height:400px;" ></div>
</body>
```

然后就可以通过 echarts. init 方法初始化一个 ECharts 实例并通过 setOption 方法生成一个简单的柱状图，以下是完整代码。

```
<! DOCTYPE html>
<html>
<head>
<meta charset = " utf-8" >
<title>ECharts</title>
<! --引入 echarts. js -->
<script src = " echarts. min. js" ></script>
</head>
<body>
<! --为 ECharts 准备一个具备大小(宽高)的 DOM -->
<div id = " main"  style = " width: 600px;height:400px;" ></div>
<script type = " text/javascript" >
        //基于准备好的 DOM,初始化 ECharts 实例
        var myChart = echarts. init( document. getElementById( 'main' ) ) ;
        //指定图表的配置项和数据
```

```
        var option = {
            title: {
                text: 'ECharts 入门示例'
            },
            tooltip: {},
            legend: {
                data:['销量']
            },
            xAxis: {
                data: ["衬衫","羊毛衫","雪纺衫","裤子","高跟鞋","袜子"]
            },
            yAxis: {},
            series: [{
                name: '销量',
                type: 'bar',
                data: [5, 20, 36, 10, 10, 20]
            }]
        };
        //使用刚指定的配置项和数据显示图表。
        myChart. setOption(option);
</script>
</body>
</html>
```

ECharts 实例效果如图 2-6-5 所示。

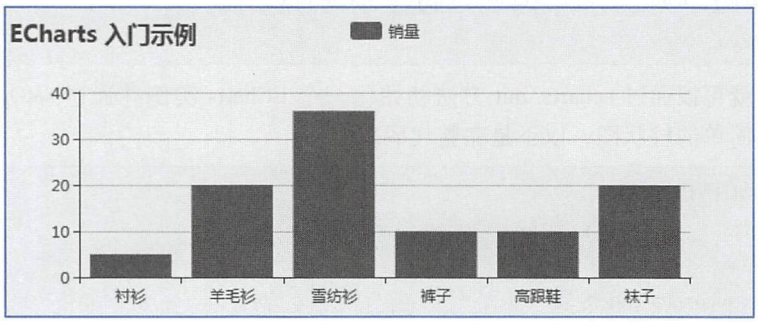


图 2-6-5　ECharts 实例效果

项目总结

在本项目中介绍了项目开发过程中用到的各种框架和技术，学生通过本项目的学习可以对相关开发技术有一个宏观的了解，包括前台网页开发框架 Bootstrap、异步通信技术 Ajax、Spring、SpringMVC、MyBatis 框架和 ECharts 图表可视化插件，这些技术和框架都会在后续的技能训练环节加强学生的认识。

项目 3　开发环境与工具介绍

PPT 开发环境与工具介绍

项目描述

本项目介绍了 Java Web 云应用开发环境及其相关工具软件，包括 Java 与 JDK、Apache Tomcat 服务器、Eclipse 集成开发环境、MySQL 数据库。

知识目标

- 了解 Java 与 JDK。
- 了解 Apache Tomcat 服务器。
- 了解 Eclipse 集成开发环境。
- 了解 MySQL 数据库。

任务列表

任务编号	任务名称	建议课时
任务 3.1	认识 Java 与 JDK	
任务 3.2	认识 Apache Tomcat 服务器	
任务 3.3	认识 Eclipse 集成开发环境	
任务 3.4	认识 MySQL 数据库	
	总计：	2 课时

任务 3.1　认识 Java 与 JDK

微课 3
开发环境与工具介绍

【任务目标】

- 了解 Java 编程语言。
- 了解 JDK 软件开发工具包。
- 熟悉 Java 运行环境 JRE。

【技术要点】

1. 面向对象编程语言 Java

Java 是一门面向对象编程语言，不仅吸收了 C++语言的各种优点，还摒弃了 C++中难以理解的多继承、指针等概念，因此 Java 语言具有功能强大和简单易用两个特征。Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，方便程序员以面向对象的思维方式进行复杂的编程。Java 具有简单性、面向对象、分布式、健壮性、安全性、平台独立与可移植性、多线程、动态性等特点。Java 可以编写桌面应用程序、Web 应用程序、分布式系统和嵌入式系统应用程序等。

2. JDK

JDK（Java Development Kit）是 Java 语言的软件开发工具包，主要用于移动设备、嵌入式设备上的 Java 应用程序。JDK 是整个 Java 开发的核心，它包含了 Java 的运行环境（JVM+Java 系统类库）和 Java 工具。JDK 包含的基本组件如下。

- javac：编译器，将源程序转成字节码。
- jar：打包工具，将相关的类文件打包成一个文件。
- javadoc：文档生成器，从源码注释中提取文档。
- jdb：debugger，查错工具。
- java：运行编译后的 Java 程序（.class 后缀的）。
- appletviewer：小程序浏览器，一种执行 HTML 文件上的 Java 小程序的 Java 浏览器。
- Javah：产生可以调用 Java 过程的 C 过程，或建立能被 Java 程序调用的 C 过程的头文件。
- Javap：Java 反汇编器，显示编译类文件中的可访问功能和数据，同时显示字节代码含义。
- Jconsole：Java 进行系统调试和监控的工具。

Java 结构图，如图 3-1-1 所示。

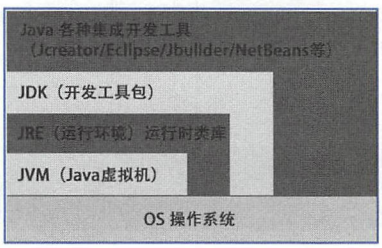


图 3-1-1　Java 结构图

3. JRE

JRE（Java Runtime Environment）Java 运行环境，包括 Java Runtime Environment 和 Java Plug-in 两部分。JRE 是可以在其上运行、测试和传输应用程序的 Java 平台。它包

括 JVM（Java 虚拟机）、Java 核心类库和支持文件，但不包含开发工具（JDK）——编译器、调试器和其他工具。JRE 需要辅助软件（Java Plug-in）以便在浏览器中运行 applet。

JRE 可以支撑 Java 程序的运行，包括 JVM 虚拟机（java. exe 等）和基本的类库（rt. jar 等），JDK 可以支持 Java 程序的开发，包括编译器（javac. exe）、开发工具（javadoc. exe、jar. exe、keytool. exe、jconsole. exe）和更多的类库（如 tools. jar）等。

任务 3.2　认识 Tomcat 服务器

【任务目标】

- 了解 Tomcat 服务器。
- 了解 Tomcat 目录组成。
- 了解常用的其他 Java Web 服务器。

【技术要点】

1. Tomcat 简介

Tomcat 是 Apache 软件基金会（Apache Software Foundation）的 Jakarta 项目中的一个核心项目，由 Apache、Sun 和其他一些公司及个人共同开发而成。它是一个开放源代码、运行 Servlet 和 JSP Web 应用软件的基于 Java 的 Web 应用软件容器，受到越来越多的软件公司和开发人员的喜爱。Tomcat 是一个完全免费的软件，任何人都可以从互联网上自由地下载，十分方便。

作为 JSP 引擎，Tomcat 服务器负责接受浏览器客户端的 Web 请求，将请求传送给 JSP Web 应用进行处理，并将处理结果（响应）返回浏览器客户端。

2. Tomcat 目录组成

Tomcat 安装目录中有 bin、webapps 等，其作用见表 3-2-1。

表 3-2-1　Tomcat 目录组成及作用

文件目录	作　用
/bin	存放所有关闭或启动服务器的可执行文件
/conf	存放服务器的配置文件，如 server. xml、web. xml 等
/lib	存放 Tomcat 服务器和所有 Web 应用程序需要访问的 JAR 文件
/logs	存放服务器日志文件
/webapps	Tomcat 的主要 Web 发布目录，默认情况下把 Web 应用文件放于此目录
/work	存放 JSP 编译后产生的 class 文件

3. Tomcat 使用举例

下面通过一个简单的例子来说明如何在 Tomcat 中发布网站，并在浏览器中进行访问。新建文件夹 TestWeb，使用记事本或其他文本编辑工具在该文件夹中创建一

个 JSP 文件 test. jsp，输入如下代码。

```
<html>
   <head><title>Test </title></head>
   <body>
<h1><% out. print( "Tomcat Test" ); %></h1>
   </body>
</html>
```

接着，进入 Tomcat 安装目录的 webapps，可以看到 root、manager、doc 等子目录，将之前创建的 TestWeb 文件夹复制到 webapps。该过程相当于将网站发布到服务器上。在浏览器中输入“http://localhost:8080/TestWeb/test. jsp”，这里一定要注意严格区分大小写。运行结果如图 3-2-1 所示。

注意

如果 Tomcat 没有启动，在浏览器中直接打开该页面文件，则该文件不能被成功执行，因为浏览器不能成功解释其中<%...%>中的代码。

图 3-2-1　JSP 页面运行界面

4. 常见的 Web 服务器

Tomcat 是一个小型的轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试 JSP 程序的首选。除了 Tomcat 之外，使用 JSP 进行 Web 开发，还可以使用其他的 Web 服务器，如 WebLogic、Jboss 等。

（1）WebLogic

WebLogic 是美国甲骨文（Oracle）公司出品的一个应用服务器，是商业市场上主要的 Java（J2EE）应用服务器软件之一。

WebLogic 全面实现了 J2EE 1.5 规范、Web 服务标准和互操作标准，其内核以可执行、可扩展和可靠的方式提供统一的安全、事务和管理服务。提供高级消息传输、数据持久性、管理、高可用性、集群和多平台开发支持。

（2）Jboss

Jboss 是一个基于 J2EE 的开放源代码的应用服务器，它是 J2EE 应用服务器领域发展较为迅速的应用服务器。Jboss 应用服务器具有以下优秀的特质：

- 具有革命性的 JMX 微内核服务作为其总线结构。

- 面向服务的架构（Service-Oriented Architecture，SOA）。
- 具有统一的类装载器，从而能够实现应用的热部署和热卸载能力。

Jboss 应用服务器健壮且高质量，具有良好的性能。为满足企业级市场日益增长的需求，Jboss 公司从 2003 年开始就推出了专业级产品支持服务。Jboss 始终紧跟最新的 J2EE 规范，而且在某些技术领域引领 J2EE 规范的开发。因此，无论在商业领域，还是在开源社区，Jboss 均成为了第一个通过 J2EE 1.4 认证的主流应用服务器。现在，Jboss 应用服务器已经真正发展成具有企业强度（即支持关键级任务的应用）的应用服务器。

（3）Jetty

Jetty 是一个开源、基于标准且具有丰富功能的 HTTP 服务器和 Web 容器。Jetty 是使用 Java 语言编写的，开发人员可以将 Jetty 容器实例化成一个对象，快速为一些独立运行的 Java 应用提供网络和 Web 连接。

Jetty 可以作为一个传统的 Web 服务器来处理静态和动态网页，或作为专用 Http 服务器的后台来处理动态网页，还可以作为一个 Java 应用程序的内嵌组件。这样的灵活性意味着它可以用在多种场合，可随产品做外盒使用，例如 Tapestry、Liferay；放在随书光盘里，用来运行例子；合并到程序里提供 HTTP 传输或集成到 JavaEE 服务器作为 Web 容器等。

（4）JRun

JRun 是一个具有广泛适用性的 Java 引擎，用于开发及实施由 Java Servlets 和 JSP 编写的服务器端 Java 应用，目前最新的版本是 JRun 4。JRun 是第一个完全支持 JSP 1.0 规格书的商业化产品，全球有超过 80 000 名开发人员使用 JRun 在他们已有的 Web 服务器（包括 Microsoft IIS、Netscape Enterprise Server、Apache 等）上添加服务器端 Java 的功能。

JRun 有开发者、专业、高级和企业版 4 个版本。各版本的特性及适用条件详见相关的参考资料。

5. Tomcat 配置虚拟目录

在 Tomcat 实例中，将 TestWeb 网站复制到了 Tomcat 安装目录的 webapps 中进行发布后浏览，也可以通过设置虚拟目录的方式浏览网站。

① 假设将 D:\TestWeb 目录配置为虚拟目录，用文本编辑软件 EditPlus 打开 Tomcat 安装目录下 Conf 文件夹中的 web.xml，它是一个配置文件，用于对 Tomcat 进行相关的配置。找到如图 3-2-2 中所示的代码段，将第 107 行中的 false 改成 true。

```
 98      <servlet>
 99          <servlet-name>default</servlet-name>
100          <servlet-class>org.apache.catalina.servlets.DefaultServlet</servlet-class>
101          <init-param>
102              <param-name>debug</param-name>
103              <param-value>0</param-value>
104          </init-param>
105          <init-param>
106              <param-name>listings</param-name>
107              <param-value>true</param-value>
108          </init-param>
109          <load-on-startup>1</load-on-startup>
110      </servlet>
```

图 3-2-2　web.xml 文件配置

② 在 Tomcat 安装目录的 Conf\Catalina\localhost 路径下创建一个 XML 文件，文件名与 Web 项目的名字保持一致，这里应该为 TestWeb.xml，然后在文件中添加以下代码：

```
<context path="/TestWeb"  docBase="D:\TestWeb"  debug="0"  reloadable="true"  cross-Context="true"></context>
```

其中，path="/TestWeb"表示配置的虚拟目录的名称，docBase="D:\TestWeb"是虚拟目录指向的事实目录。启动 Tomcat，访问 http://localhost:8080/TestWeb/test.jsp 可以浏览到与图 3-2-1 相同的页面。

任务 3.3　认识 Eclipse 集成开发环境

【任务目标】

- 了解 Eclipse。
- 了解 Eclipse 各种版本。
- 了解其他常用的 Java 集成开发环境。

【技术要点】

1. Eclipse 简介

Eclipse 最初是由 IBM 公司开发的替代商业软件 Visual Age for Java 的下一代 IDE 开发环境，2001 年 11 月贡献给开源社区，现在它由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。Eclipse 项目由 IBM 公司发起，围绕着 Eclipse 项目已经发展成为了一个庞大的 Eclipse 联盟，有 150 多家软件公司参与到 Eclipse 项目中，其中包括 Borland、Rational Software、Red Hat 及 Sybase 等公司。Eclipse 是一个开放源码项目，由于其开放源码，任何人都可以免费得到，并可以在此基础上开发各自的插件，因此越来越受人们关注。随后还有包括甲骨文在内的许多大公司也纷纷加入了该项目，Eclipse 的目标是成为可进行任何语言开发的 IDE 集成者，使用者只需下载各种语言的插件即可。

2. Eclipse 体系结构

Eclipse 是一种通用的工具平台，不仅可以用来开发 Java 程序，也可以用来开发 C/C++、PHP 等程序，而且是一种开放式扩展的集成开发环境，即任何人都可以扩展 Eclipse 的功能，Eclipse 体系结构图如图 3-3-1 所示。

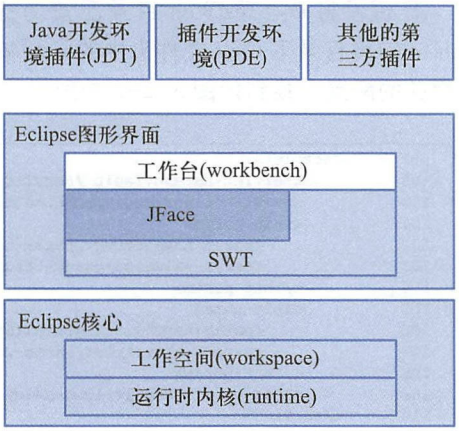


图 3-3-1　Eclipse 体系结构图

Eclipse 的设计思想是：一切皆插件。Eclipse 核心很小，其他所有功能都

以插件的形式附加于 Eclipse 核心之上。Eclipse 基本内核包括图形 API（SWT/Jface）、Java 开发环境插件（JDT）、插件开发环境（PDE）等。Eclipse 的插件机制是轻型软件组件化架构，Eclipse 对这些插件的协同工作提供了良好的支持，不仅安装简单，且可以无缝结合。Eclipse 对这些插件是动态载入并动态调用的。所谓动态是指 Eclipse 启动后要用到某个插件时，该插件才被调入内存，而当插件不再被使用时，它会在适当的时候被清除出内存。因此不必担心不常用的插件会消耗内存。

在客户机平台上，Eclipse 使用插件来提供所有的附加功能，如支持 Java 以外的其他语言。已有的分离的插件已经能够支持 C/C++（CDT）、Perl、Ruby，Python、telnet 和数据库开发。插件架构能够支持将任意的扩展加入到现有环境中，例如配置管理，而决不仅仅限于支持各种编程语言。

3. Eclipse 的版本发布

Eclipse 发展至今经历了以下几个版本。

2001 年 11 月由 IBM 公司贡献给开源社区，现在它由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理，Eclipse 1.0 发布。

2002 年 6 月 27 日 Eclipse 进入了 2.0 时代。2.0 时代的 Eclipse 经历了 2.0 和 2.1 两个大的版本。其中 2.0 在之后又推出了 2.0.1 和 2.0.2 版本。

2003 年 3 月 27 日推出的 2.1 版本，也接连推出了 2.1.1、2.1.2 和 2.1.3 三个后续修订版本。

2004 年 6 月 25 日对 Eclipse 来说是一个值得纪念的日子，进入 3.0 时代的 Eclipse 采用了 OSGi 运行时架构。这一年 Eclipse 基金会成立，这也标志着 Eclipse 进入一个新的时代。3.0 后有两个小的修订版本分别是 3.0.1 和 3.0.2。

2005 年 6 月 27 日 Eclipse 3.1 发布，之所以要特别提一下 3.1，是因为从这个版本开始一直到还没有发布的 3.5 版，形成了一个以木星卫星名称相关的系列。1610 年，著名的科学家伽利略通过对木卫 1～木卫 4 的观察，提出了反驳地心说的证据，木卫 1～木卫 4 因此也被称之为伽利略四大卫星（木星还有很多之后发现的其他卫星）。这 4 大卫星还被分别赋予了 4 个神话传说中人物的名字，如木卫 1：IO，伊奥；木卫 2：Europa，欧罗巴；木卫 3：Ganymede，盖尼米德；木卫 4：Callisto，卡里斯托。

Eclipse 从 3.1～3.4 的命名（codename）并非按照木卫 1～木卫 4 的顺序，而是根据这 4 颗卫星距离木星从近到远的顺序，因此 Eclipse 3.1 就使用了木星已知卫星中第 1 近的木卫——IO 来命名，Eclipse 3.2 就使用木卫 4——Callisto。

表 3-3-1 列出了已知的版本代号。

表 3-3-1　Eclipse 版本代号

版 本 号	时　间	版本代号
Eclipse 1.0	2001 年 11 月 7 日	
Eclipse 2.0	2002 年 6 月 27 日	
Eclipse 2.1	2003 年 3 月 27 日	
Eclipse 3.0	2004 年 6 月 25 日	

续表

版 本 号	时　间	版本代号
Eclipse 3.1	2005年6月27日	IO（木卫1，伊奥）
Eclipse 3.2	2006年6月26日	Callisto（木卫4，卡里斯托）
Eclipse 3.3	2007年6月27日	Eruopa（木卫2，欧罗巴）
Eclipse 3.4	2008年6月25日	Ganymede（木卫3，盖尼米德）
Eclipse 3.5	2009年6月24日	Galileo（伽利略）
Eclipse 3.6	2010年6月23日	Helios（太阳神）
Eclipse 3.7	2011年6月22日	Indigo（靛青）
Eclipse 3.8/4.2	2012年6月27日	Juno（婚神星）
Eclipse 4.3	2013年6月26日	Kepler（开普勒）
Eclipse 4.4	2014年6月25日	Luna（月神）
Eclipse 4.5	2015年6月25日	Mars（火星）
Eclipse 4.6	2016年6月25日	Neon（霓虹灯）
Eclipse 4.7	2017年5月25日	Oxygen（氧气）

4. 其他常用 Java 集成开发环境

Eclipse 因其开源免费、使用方便，而且有众多插件可以集成等特点，赢得了众多 Java 开发人员的青睐，是目前非常流行的 Java 集成开发环境软件之一，除此之外，还有 IntelliJ IDEA、MyEclipse 等。

（1）IntelliJ IDEA

IntelliJ IDEA 简称 IDEA，在业界被公认为最好的 Java 开发工具之一，尤其在智能代码助手、代码自动提示、重构、J2EE 支持、各类版本工具（git、svn、github 等）、JUnit、CVS 整合、代码分析、创新的 GUI 设计等方面。

IDEA 最突出的功能是调试（Debug），可以对 Java 代码、JavaScript、jQuery、Ajax 等技术进行调试。这点甚至超越 Eclipse。除此之外其智能选取功能、丰富导航模式、编码辅助、代码自动检查、支持 Java EE 开发、智能代码等功能大大提高了程序员的编程效率。而其缺点是，太过强大的自动提示会让编程者逐渐依赖于该软件，尤其对于初学者而言各有利弊。

（2）MyEclipse

MyEclipse 是一个十分优秀的用于开发 Java、J2EE 的商业软件，它可以看作是 Eclipse 插件集合，其功能非常强大，支持也十分广泛，尤其是对各种开源产品的支持较好。MyEclipse 可以支持 Java Servlet、Ajax、JSP、JSF、Struts、Spring、Hibernate、EJB3、JDBC 数据库链接工具等多项功能。可以说 MyEclipse 是几乎囊括了目前所有主流开源产品的专属 Eclipse 开发工具。

MyEclipse 在使用和外观上和 Eclipse 非常相似，其最大优点是集成了众多插件，方便程序员的开发，而其缺点是付费、内存消耗较大。

任务 3.4　认识 MySQL 数据库

【任务目标】

- 了解 MySQL 数据库。
- 了解 MySQL 数据库与同类型数据库的优缺点。
- 了解 MySQL 数据库的图形化管理工具。

【技术要点】

1. MySQL 简介

MySQL 是一个关系型数据库管理系统，由 MySQL AB 公司开发，目前属于甲骨文公司。MySQL 是最流行的关系型数据库管理系统，在 Web 应用方面 MySQL 是最好的关系数据库管理系统（Relational Database Management System，RDBMS）应用软件之一。MySQL 是一种关联数据库管理系统，关联数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就提高了速度和灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的常用标准化语言。MySQL 软件采用了双授权政策，它分为社区版和商业版，因其具有体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

2. MySQL、SQL Server、Oracle 的对比

(1) MySQL

优点是支持 5000 万条记录的数据仓库；适应于所有的平台；是开源软件，版本更新较快；性能很出色，线程是轻量级的进程，可灵活地为用户提供服务并且价格便宜。

缺点是缺乏一些存储程序的功能，如 MyISAM 引擎联支持交换功能。对存储过程和触发器支持不够良好。

(2) Microsoft SQL Server

优点是真正的客户机/服务器体系结构；图形化的用户界面，使系统管理和数据库管理更加直观、简单；丰富的编程接口工具，为用户进行程序设计提供了更大的选择余地；与 WinNT 完全集成，利用了 NT 的许多功能，如发送和接收消息，管理登录安全性等，SQL Server 也可以很好地与 Microsoft BackOffice 产品集成；同时缺点也客观存在：开放性，只能运行在微软的 Windows 平台，没有丝毫的开放性可言；可伸缩性和并行性，SQL Server 并行实施和共存模型并不成熟，很难处理日益增多的用户数和数据卷，伸缩性有限；性能稳定性，SQL Server 当用户连接多时性能会变得很差，并且不够稳定；使用风险，SQL Server 并不十分兼容早期产品，使用需要冒一定风险；客户端支持及应用模式，只支持 C/S 模式。

(3) Oracle

优点是 Oracle 的稳定性要比 SQL Server、My SQL 好；Oracle 的导数据工具 sqlload. exe 功能比 SQL Server 的 Bcp 功能强大，Oracle 可以按照条件把文本文件数据导

入；Oracle 的安全机制更好；在处理大数据方面 Oracle 会更稳定一些；SQL Server 在数据导出方面功能更强一些。

缺点是对硬件的要求很高；价格比较昂贵；管理维护麻烦一些；操作比较复杂，技术含量较高。

3. MySQL 数据库的图形化管理工具

MySQL 数据库的界面是命令行界面，如图 3-4-1 所示。对于熟悉 DOS 命令行操作或者 Linux 命令行操作的读者来说，可能不需要图形化管理工具也可以熟练操作，然而对于大多数初学者而言，没有图形化管理工具会使得数据库操作变得复杂，以下是几种常用的 MySQL 图形化管理工具。

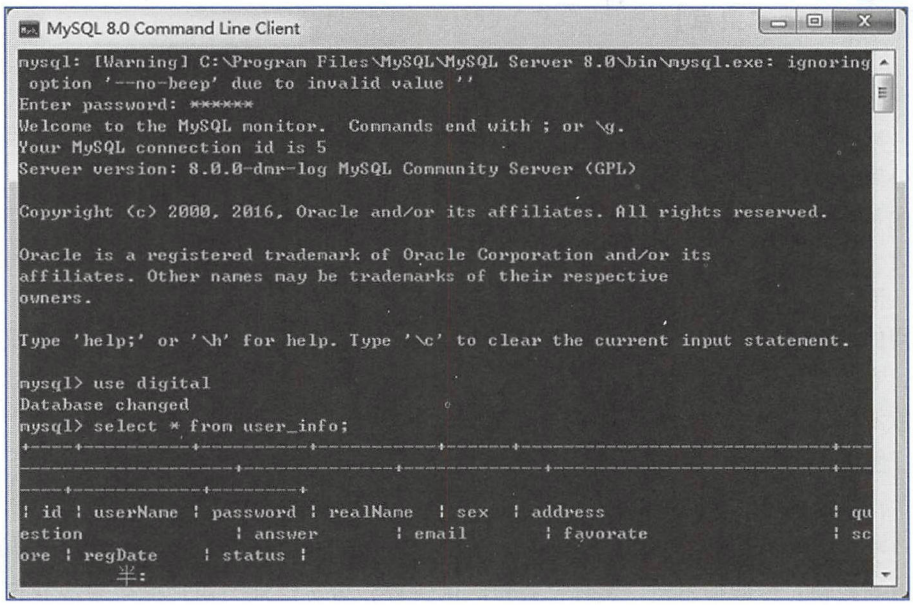


图 3-4-1　MySQL 数据库的界面是命令行界面

(1) MySQL Workbench

MySQL Workbench 是一个开源的统一的可视化开发和管理平台，该平台提供了许多高级工具，可支持数据库建模和设计、查询开发和测试、服务器配置和监视、用户和安全管理、备份和恢复自动化、审计数据检查以及向导驱动的数据库迁移。MySQL Workbench 为数据库管理员、程序开发者和系统规划师提供可视化设计、模型建立、以及数据库管理功能。它包含了用于创建复杂的数据建模 ER 模型，正向和逆向数据库工程，也可以用于执行通常需要花费大量时间和难以变更和管理的文档任务。MySQL 工作台可在 Windows、Linux 和 Mac 上使用。

(2) Navicat

Navicat 是一个桌面版 MySQL 数据库管理和开发工具。和微软 SQLServer 的管理器很像，易学易用。Navicat 使用图形化的用户界面，可以让用户使用和管理更为轻松。其支持中文，但收费。

(3) SQLyog

SQLyog 是一个易于使用的、快速而简洁的图形化管理 MySQL 数据库的工具，

它能够在任何地点有效地管理用户的数据库。SQLyog 是业界著名的 Webyog 公司出品的一款简洁高效、功能强大的图形化 MySQL 数据库管理工具。使用 SQLyog 可以快速直观地从世界的任何角落通过网络来维护远端的 MySQL 数据库。

项目总结

了解开发环境和选择开发工具是开发前的重要准备工作，本项目对开发必备环境进行了一个宏观的讲解，并对需要用到的工具及其同类型的工具产品进行了介绍。学生可以根据个人需求及偏好选择合适的开发工具。

第2篇 技能训练篇

项目4 系统概要设计

PPT 系统概要设计

项目描述

数据，已经渗透到当今每一个行业和业务职能领域，成为重要的生产因素，而数据可视化部分是数据分析的入口与展示，起着非常重要的作用。 本书选取了“新能源汽车智能监控管理系统”作为教学项目，结合相关知识点详细讲解了项目的实现过程，在本项目中重点介绍了“新能源汽车智能监控管理系统”项目的背景知识，为后面的学习作好铺垫。

知识目标

- 熟悉 Web 项目需求分析。
- 理解概要设计的目的和主要方法。
- 熟悉系统详细设计的方法。
- 熟悉 Web 项目的数据库设计。

技能目标

- 能理解项目需求，按照规范完成项目需求分析。
- 能运用工具进行概要设计项目系统分析。
- 能完成系统各模块详细设计。
- 能熟练完成数据库设计。

任务列表

任务编号	任务名称	建议课时
任务 4.1	分析系统需求和实现功能设计	0.5
任务 4.2	详细设计系统	0.5
任务 4.3	设计数据库	0.5
技能训练	车友圈模块功能的用例图和类图	0.5
	总计：	2 课时

任务 4.1　分析系统需求和实现功能设计

【任务描述】

本任务主要介绍新能源汽车的主要功能和网站模块划分。

微课 4
新能源汽车智能监控管理系统概要设计

【任务目标】

知识目标

- 熟悉需求分析的方法。
- 熟悉 Web 系统功能设计的方法。
- 熟悉原型设计方法。

技能目标

- 能正确完成系统需求分析。
- 能够根据需求分析完成功能设计。
- 能够根据功能设计要求完成原型设计。

【任务分析】

要完成新能源汽车智能监控管理系统，首先要进行充分的需求分析，其次根据需求分析确定系统功能模块，最后完成原型设计，为开发做好准备工作。

【任务实施】

1. 系统需求分析

新能源汽车智能监控管理系统是一个以 Java Web 为核心开发技术的应用系统，该 Web 应用可以对汽车的监控数据进行数据可视化显示，用户可以非常直观地监控到有关汽车的性能指标、汽车内部零部件的指标，以体现智能制造并更好地为企业服务。新能源汽车的实时数据的发送和接收部分按照行业标准封装完成，由于涉及汽车部分数据的行业保密性，此部分在本书中不做阐述，仅对数据接收存储后的数据管理和监控显示进行详细阐述，包括 Java Web 框架编程开发、数据可视化技术以及云平台搭建和部署等重要内容。

2. 系统功能分析

新能源汽车智能监控管理系统主要包括包含普通车主和企业管理员两类用户角色。普通车主用户的功能包括用户注册、用户登录、车辆分布、车友圈、地图导航、个人车辆管理等操作；企业管理员的功能包括用户登录、用户管理、企业用户管理、企业车辆管理、车辆监控等相关操作。网站模块结构图如图 4-1-1 所示。

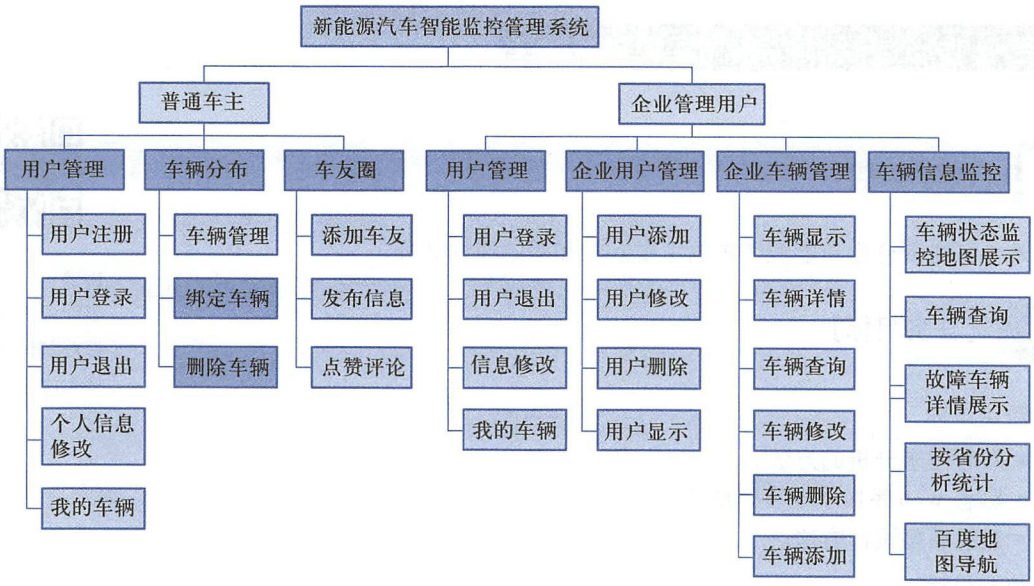


图 4-1-1　网站模块结构图

(1) 用户管理

普通车主用户与企业用户均有用户管理功能，主要包括以下部分。

- 用户登录。用户进入系统后，车友圈信息查看与发布、车辆的绑定与删除等操作，需进行身份验证后才可进入。已注册的用户可通过输入用户名、密码、验证码进入网站。
- 用户注册。首次进入网站的新用户可通过用户注册模块，填写用户相关的信息，其后标记“*”的项为必填项；填写完信息后，单击“确定”按钮，成为系统用户后，即可登录。
- 用户退出。当用户单击“退出登录”按钮后不可查看个人及车辆相关信息。
- 个人信息修改。用户登录后可对个人注册时填入的信息进行修改。
- 我的车辆。可以在线绑定自己的车辆信息，并查看车辆相关的行程、实时故障、故障分析等信息。

(2) 车辆分布

- 车辆分布：可在地图上查看各省份车辆分布情况，并可按省份或者按车辆状态情况分别进行查询。
- 车辆管理：可对车辆进行绑定与解绑。

(3) 车友圈

- 提供车友进行交流的平台。
- 添加车友：可以通过车友名称查找并添加车友。
- 发布信息：可以在车友圈发布信息，仅添加为车友的用户才可以查看到发布的信息。
- 点赞评论：可以对车友发布的信息进行点赞和评论。

(4) 地图导航

提供车友进行交流的平台，可根据出发地和目的地进行导航查询和显示。

(5) 企业用户管理

企业用户除了拥有车主用户的所有功能外，还具有用户管理功能。

- 用户显示：将当前系统中所有的车主信息列表显示出来。
- 用户添加：添加一个新的用户。
- 用户修改：可对已有的用户信息进行修改。
- 用户删除：将选择的用户进行删除。

(6) 企业车辆管理

- 车辆显示：将当前系统中所有的车辆信息列表显示，并进行分页。
- 车辆详情：当选择车辆后可进入车辆详情查看，包括车辆行程轨迹、车辆实时信息、车辆故障提醒等。
- 车辆查询：用户输入关键字后，可进行车辆的模糊查询。
- 车辆修改：可对车辆信息进行修改。
- 车辆删除：可将指定车辆信息删除。

3. 车辆监控及故障相关参数介绍

系统主要突出车辆监控和故障数据分析两个功能，其中车辆监控功能细分如下：

① 车辆基本信息，包括品牌、型号、车辆类型（混合动力汽车（HEV）、纯电动汽车（BEV）、燃料电池汽车（FCEV））、运行总里程（2871 km）等。

② 车辆基本参数信息，见表 4-1-1。

表 4-1-1　车辆基本参数信息

参数名称	参数取值示例	参数取值范围
车速	18 km/h	0 km/h ~ 255/km/h
充电状态	01	00 未连接 01 单枪默认 10 双枪快充
车辆速度	168 km/h	0 ~ 2200 km/h
累计里程	1000 km	0 ~ 9999999 km
总电压	476 V	0 ~ 10000 V
总电流	114 A	0 ~ 20000 A
加速踏板开度	0%	0% ~ 100%
制动踏板开度	0%	0% ~ 100%
前气压值	0. 89 MPa	0. 00 MPa ~ 2. 55 MPa
后气压值	0. 90 MPa	0. 00 MPa ~ 2. 55 MPa
驻气压值	0. 91 MPa	0. 00 MPa ~ 2. 55 MPa
辅助气泵气压值	0. 92 MPa	0. 00 MPa ~ 2. 55 MPa
车载电池电压	27. 4 V	0. 0 MPa ~ 51. 0 V

③ 电池参数信息，见表4-1-2。

表4-1-2　电池参数信息

参数名称	参数取值示例	参数取值范围
电池电压	595.89 V	0 V~1000 V
电池放/充电电流	110.000 A	-1500 A~1500 A
电池健康状态	100%	0%~100%
电池SOC	90%	0%~100%
单体最高电压	3.313 V	2.8 V~3.75 V
单体最高电压的单体编号	11	0~200
单体最高电压的单体箱号	1	0~200
单体最低电压	3.307 V	2.8 V~3.75 V
单体最低电压的单体编号	140	0~200
单体最低电压的单体箱号	4	0~200
电池监测点最低温度	25℃	-5℃~55℃

④ 驱动电机参数信息，见表4-1-3。

表4-1-3　驱动电机参数信息

驱动机	参数名称	参数值
驱动电机1	状态	运行，后退
驱动电机1	温度	52 ℃
驱动电机1	控制器温度	48℃
驱动电机1	转矩	84 NM
驱动电机1	转速	0 rpm
驱动电机1	电流	0.0000 A
驱动电机1	电压	587.895 V
驱动电机2	状态	停机
驱动电机2	温度	—℃
驱动电机2	控制器温度	—℃
驱动电机2	转矩	—NM
驱动电机2	转速	—rpm
驱动电机2	电流	—A
驱动电机2	电压	—V

⑤ 充电状态，见表4-1-4。

表 4-1-4　充 电 状 态

充 电 状 态	当前未充电
充电开始时间	2017/05/11 01 :45 :16
充电结束时间	2017/05/11 03 :31 :47
充电桩插入时间	2017/05/11 01 :45 :16
充电桩拔出时间	2017/05/11 04 :45 :49

4. 原型界面设计

分析了用户的功能结构后，就可以对项目进行界面设计了，本项目的主要操作界面如下。

① 首页：用户可以输入网址进入新能源汽车系统。

② 车辆分布显示。进入首页后，系统会在页面中间呈现一个地图，展现车企中的所有车辆的实时 GPS 位置，鼠标单击到每个省份时会有相应的数字及图表展示，如图 4-1-2 所示。

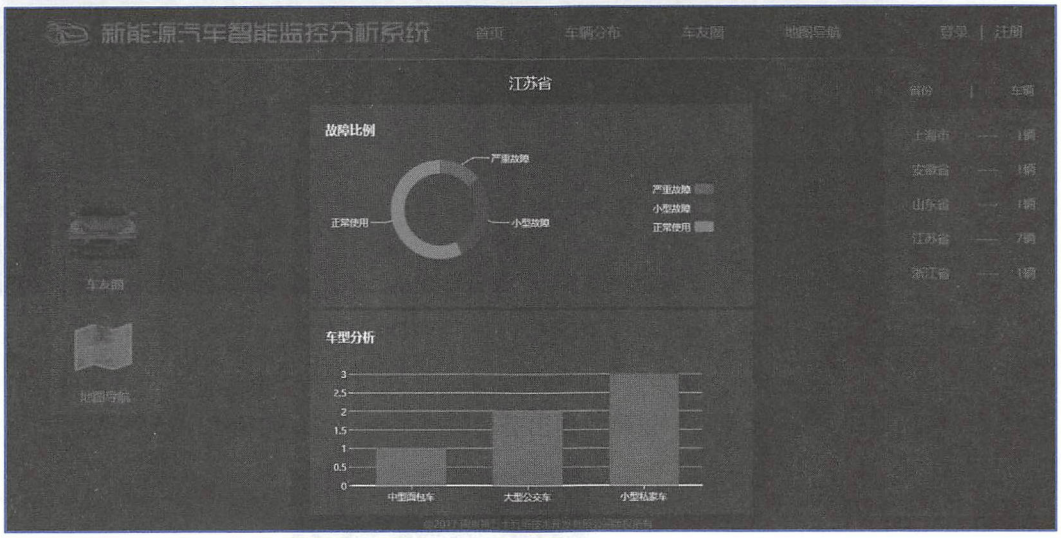


图 4-1-2　省份车辆图表

③ 详细车辆分布。该页面会展现所有汽车的分布图，可以通过筛选条件查询地图，地图上会显示车辆的状态，右侧会显示故障的情况、以及综合分析车辆的情况。

④“地图导航”界面。单击“地图导航”按钮，会跳转到“地图导航”页面，该页面显示地理位置详细地图。

⑤ 用户登录。用户可以使用账号和密码登录新能源汽车系统，如图 4-1-3 所示。

⑥“车友圈”界面。登录成功后用户就可以访问车友圈了，“车友圈”页面需要用户登录后才可以访问，显示效果如图 4-1-4 所示。

图 4-1-3　“登录”界面

图 4-1-4　“车友圈”界面

⑦ 个人中心。用户登录后会在右上角显示用户的头像并显示“基本信息”“我的车辆”“企业管理”以及“退出登录”等多个按钮，如图 4-1-5 所示。

图 4-1-5　用户头像及功能按钮

用户可以通过单击用户头像处的“基本信息”按钮进入基本信息页面，如图 4-1-6 所示。

图 4-1-6　“基本信息”页面

用户可以通过单击用户头像处的“我的车辆”按钮进入“个人车辆管理”页面，如图 4-1-7 所示。

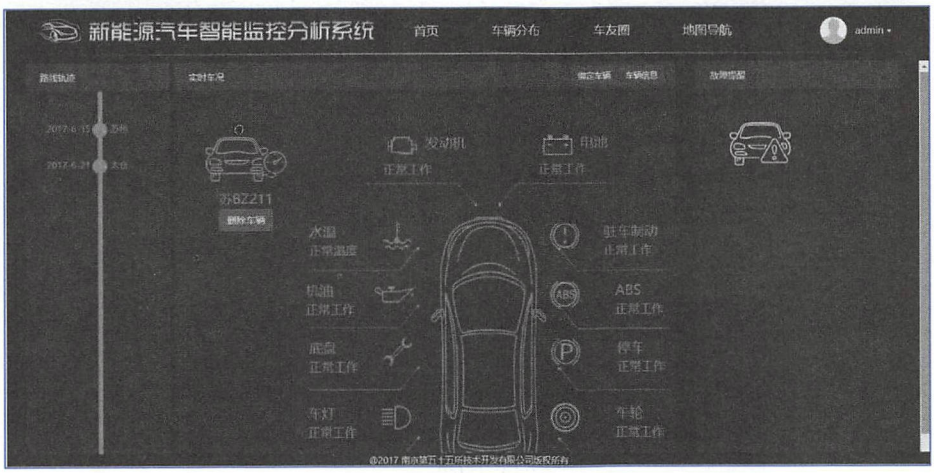


图 4-1-7　“个人车辆管理”页面

⑧ 企业界面。“企业用户管理”界面如图 4-1-8 所示。“企业车辆管理”界面如图 4-1-9 所示。

图 4-1-8　“企业用户管理”界面

图 4-1-9　“企业车辆管理”界面

5. 技术架构设计

本项目采用的是 Spring+SpringMVC+MyBatis 框架，其架构设计如下。

- 显示层使用 Bootstrap 技术搭建界面。
- 业务层使用 Spring 技术实现业务逻辑。
- 持久层使用 MyBatis 技术实现数据库联络。
- 表现层使用 Spring 技术负责具体的业务模块流程控制。

新能源汽车项目的整体技术架构如图 4-1-10 所示。

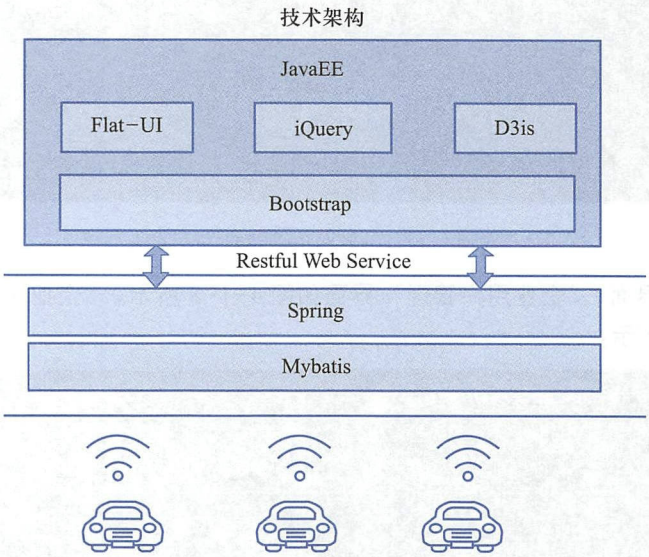


图 4-1-10　新能源汽车项目的整体技术架构

任务 4.2　详细设计系统

【任务描述】

本任务主要介绍新能源汽车的概要设计。

【任务目标】

知识目标

- 熟悉用例图的画法。
- 熟悉系统序列图的绘制方法。
- 熟悉系统类图绘制设计。

技能目标

- 能正确绘制系统各功能模块的用例图。
- 能正确绘制系统各功能模块的序列图。
- 能正确绘制系统各功能模块的类图。

【任务分析】

系统是由 Web 服务器、数据服务器和浏览器客户端组成的多层 Web 计算机服务系统，采用 Servlet-JSP-JavaBean 架构，具有灵活性、可扩展性等特点。

【任务实现】

1. 用例图

通过对本系统的相关参与者的活动进行分析和总结，本系统设计角色有两类，分别为车主用户和企业管理员，整个系统的顶层用例分析如图 4-2-1 所示。

2. 类图

类图表示不同的实体如何彼此相关，即它显示了系统的静态结构。本系统的各实体类之间的关系结构图如图 4-2-2 所示。

3. 序列图

序列图显示具体用例（或者是用例的一部分）的详细流程。它几乎是自描述的，并且显示了流程中不同对象之间的调用关系，同时还可以很详细地显示对不同对象的不同调用。

① 用户登录时序列图，如图 4-2-3 所示。

② 用户注册序列图，如图 4-2-4 所示。

③ 用户车辆绑定序列图，如图 4-2-5 所示。

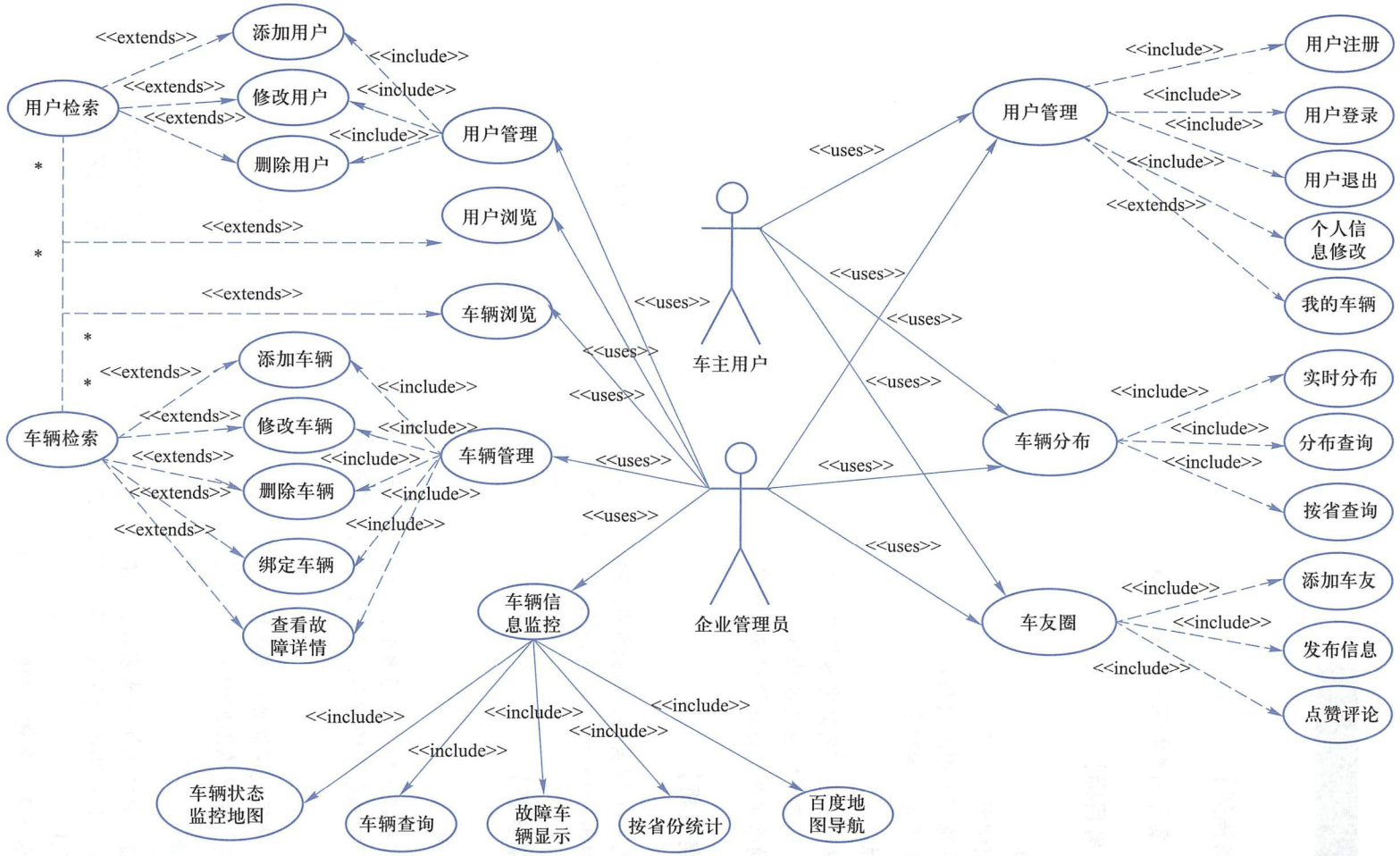

用户检索
添加用户
修改用户
删除用户
用户管理
用户浏览
车辆浏览
车辆检索
添加车辆
修改车辆
删除车辆
绑定车辆
查看故障详情
车辆管理
车辆信息监控
车辆状态监控地图
车辆查询
故障车辆显示
按省份统计
百度地图导航
车主用户
企业管理员
用户管理
用户注册
用户登录
用户退出
个人信息修改
我的车辆
车辆分布
实时分布
分布查询
按省查询
车友圈
添加车友
发布信息
点赞评论
<<extends>>
<<include>>
<<uses>>
*

图4-2-1　用例图

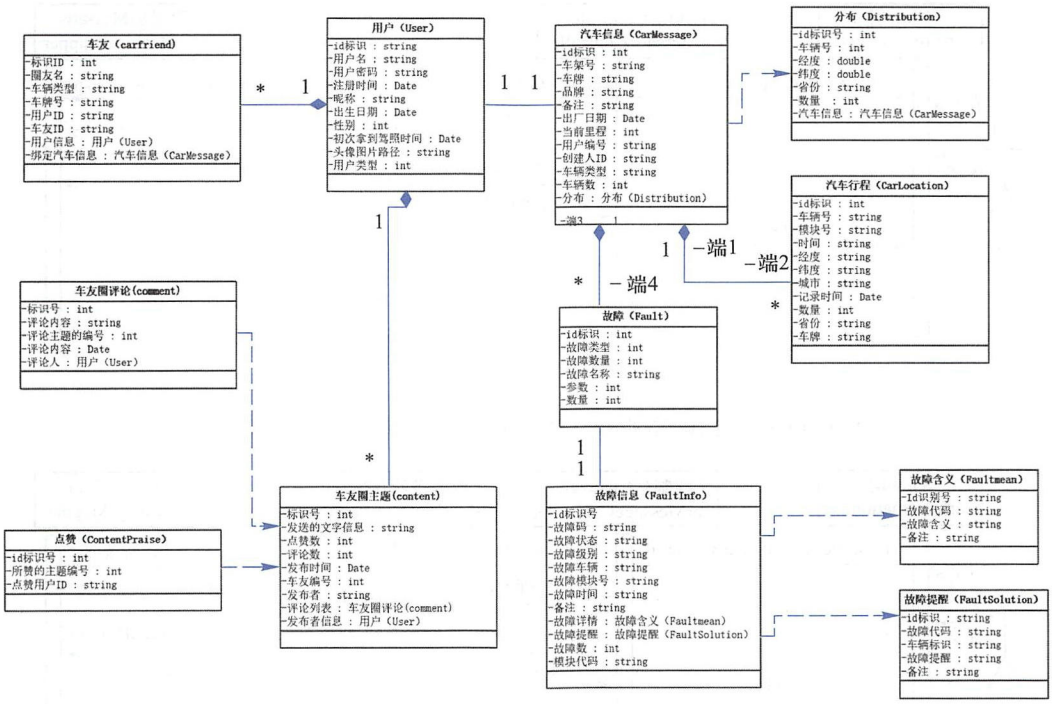


图 4-2-2　类结构图

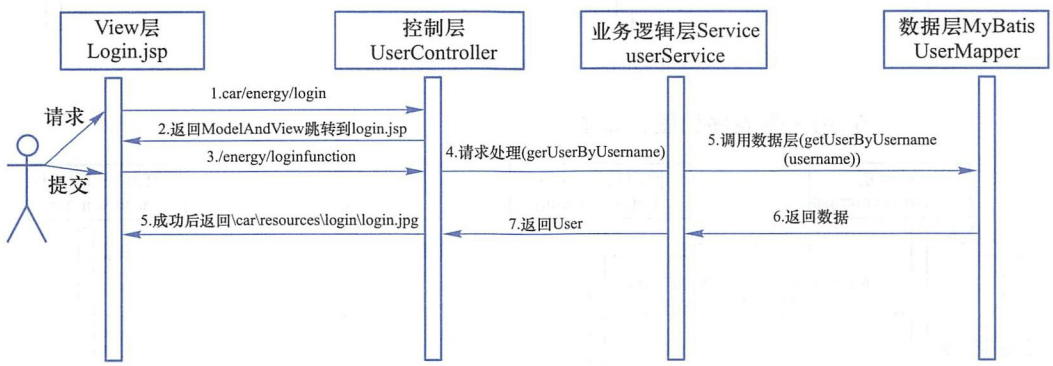


图 4-2-3　用户登录序列图

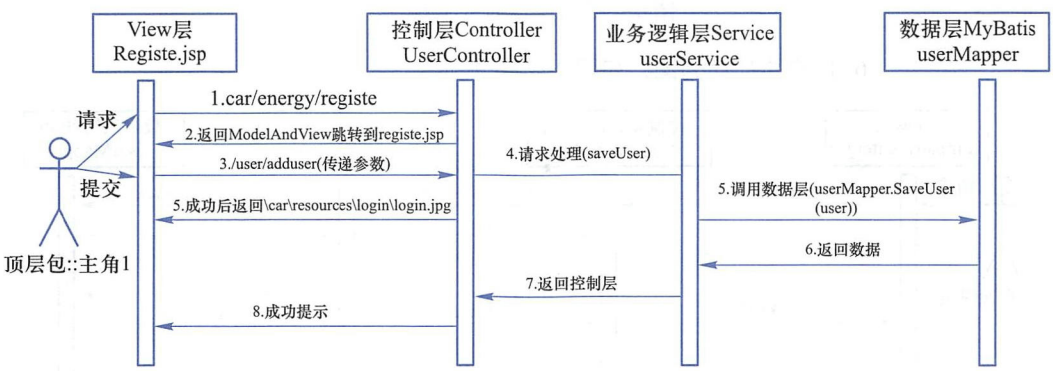


图 4-2-4　用户注册序列图

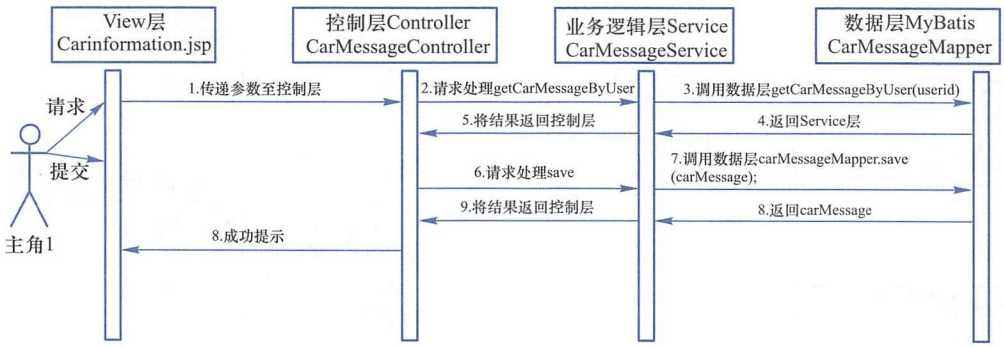


图 4-2-5　用户车辆绑定序列图

④ 用户解绑车辆序列图，如图 4-2-6 所示。

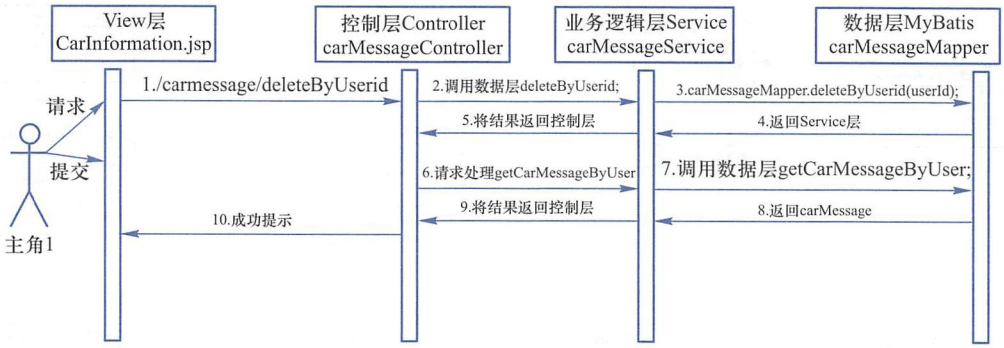


图 4-2-6　用户解绑车辆序列图

⑤ 用户列表序列图，如图 4-2-7 所示。

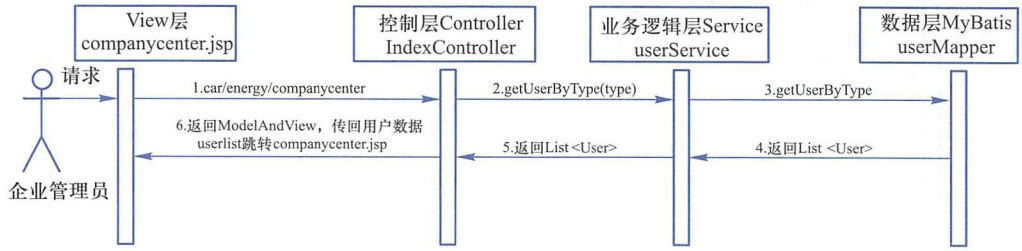


图 4-2-7　用户列表序列图

⑥ 用户添加序列图，如图 4-2-8 所示。

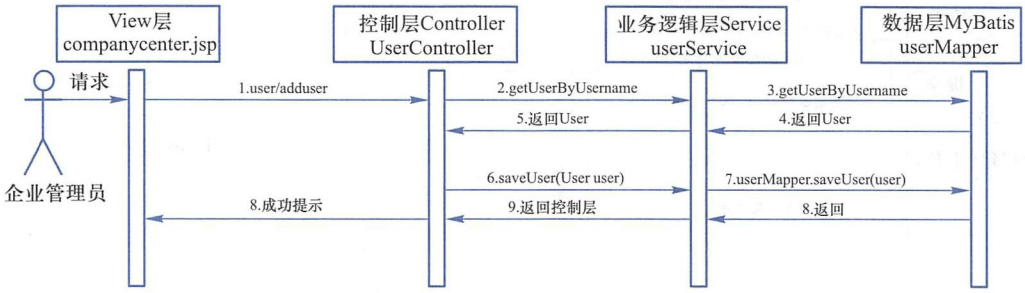


图 4-2-8　用户添加序列图

⑦ 用户删除序列图，如图 4-2-9 所示。

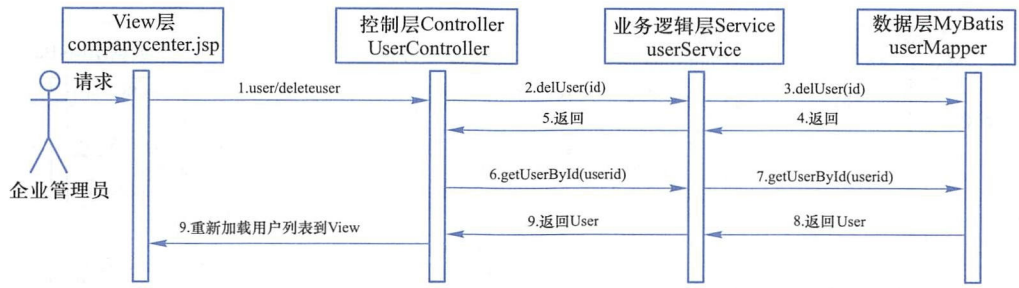


图 4-2-9　用户删除序列图

⑧ 用户修改序列图，如图 4-2-10 所示。

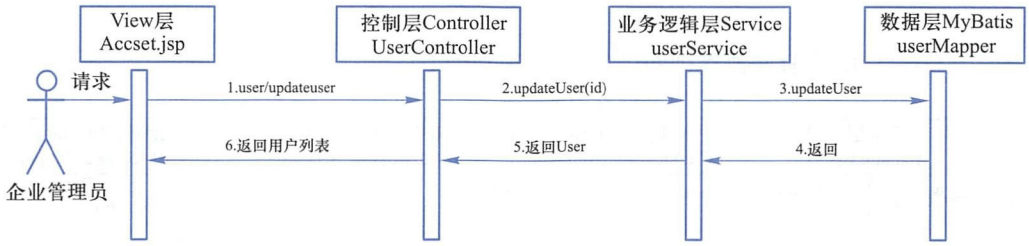


图 4-2-10　用户修改序列图

⑨ 车辆列表序列图，如图 4-2-11 所示。

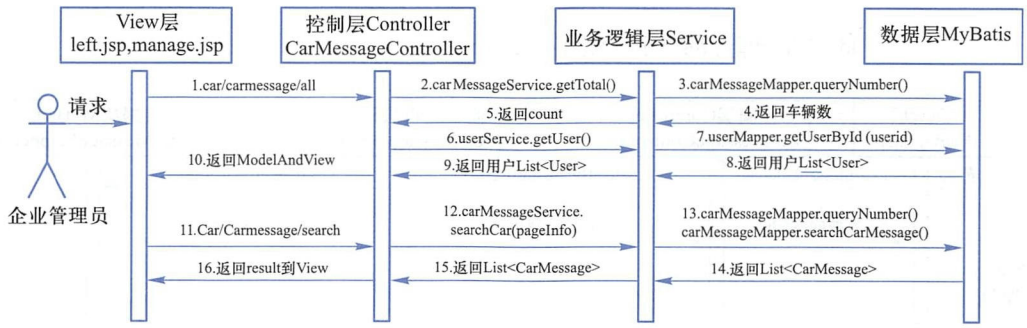


图 4-2-11　车辆列表序列图

⑩ 车辆添加序列图，如图 4-2-12 所示。

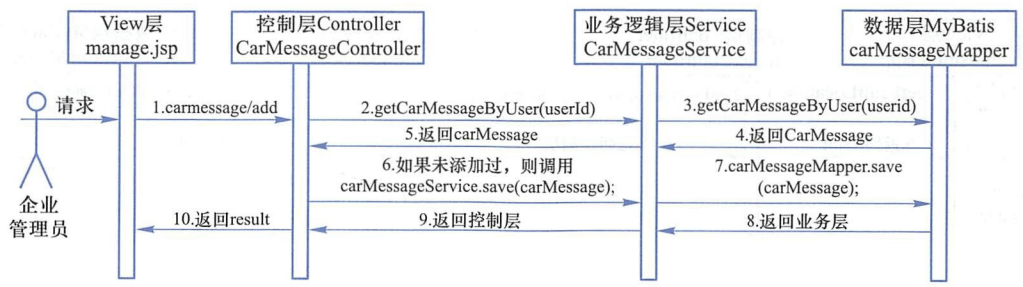


图 4-2-12　车辆添加序列图

⑪ 车辆详情序列图，如图 4-2-13 所示。

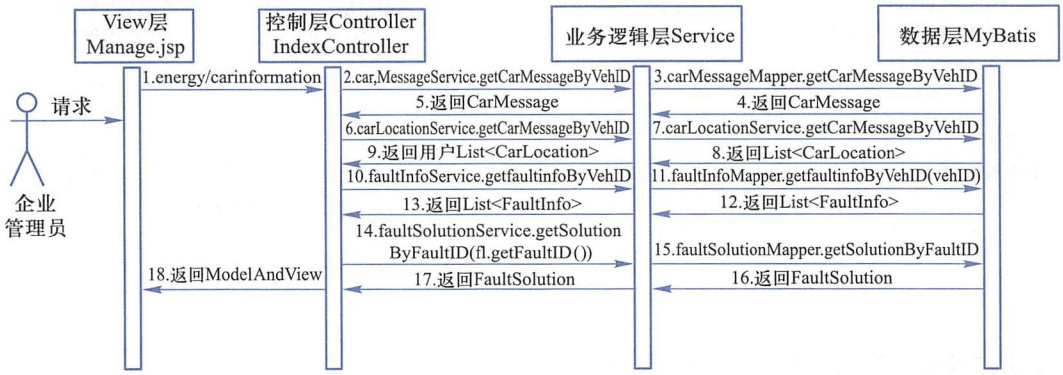


图 4-2-13　车辆详情序列图

⑫ 车辆修改序列图，如图 4-2-14 所示。

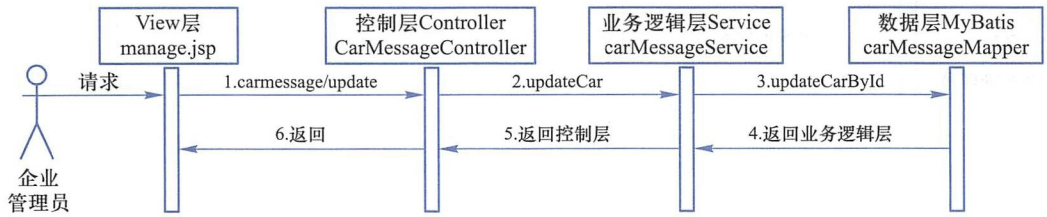


图 4-2-14　车辆修改序列图

⑬ 车辆删除序列图，如图 4-2-15 所示。

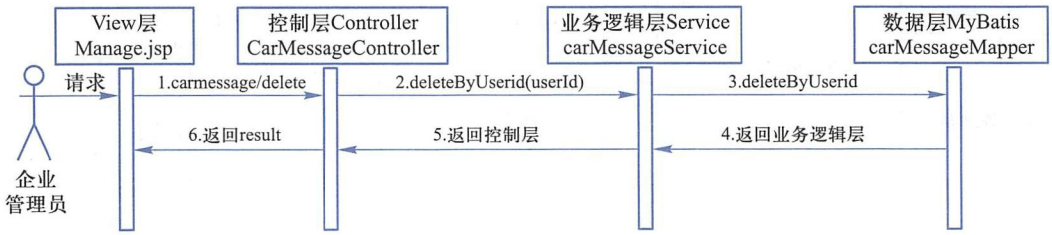


图 4-2-15　车辆删除序列图

⑭ 车辆状态监控地图展示序列图，如图 4-2-16 所示。

图 4-2-16　车辆状态监控地图展示序列图

⑮ 车辆查询序列图，如图 4-2-17 所示。

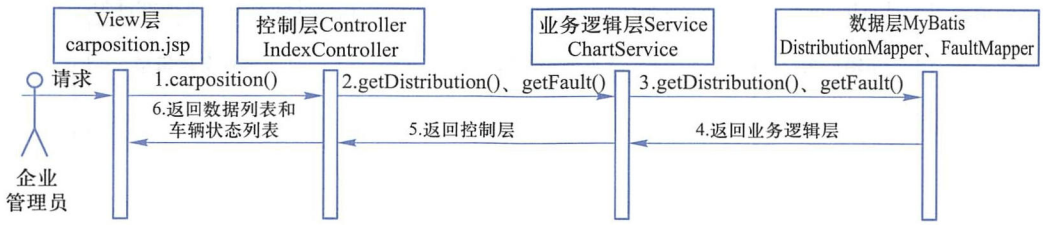


图 4-2-17　车辆查询序列图

⑯ 故障车辆统计序列图，如图 4-2-18 所示。

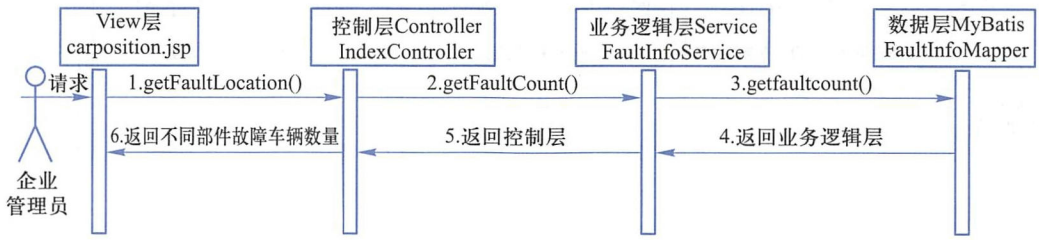


图 4-2-18　故障车辆统计序列图

⑰ 故障车辆详情显示序列图，如图 4-2-19 所示。

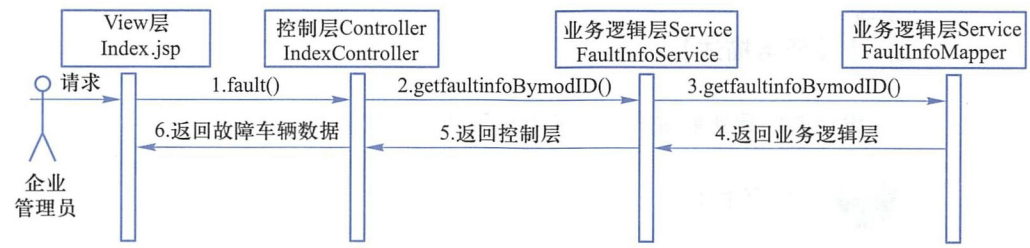


图 4-2-19　故障车辆详情显示序列图

⑱ 各省份车辆分布统计序列图，如图 4-2-20 所示。

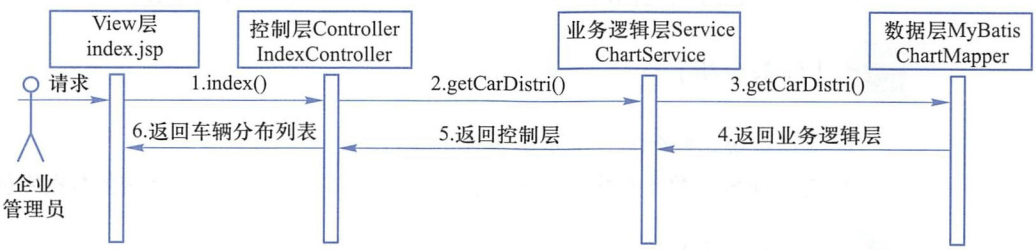


图 4-2-20　各省份车辆分布统计序列图

⑲ 故障车辆显示序列图，如图 4-2-21 所示。

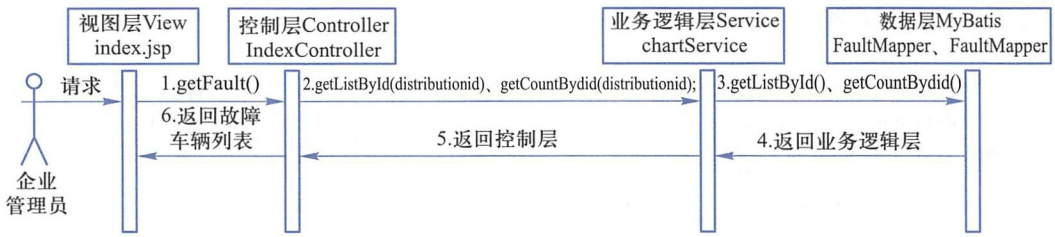


图 4-2-21　故障车辆显示序列图

⑳ 不同车型车辆数显示序列图，如图 4-2-22 所示。

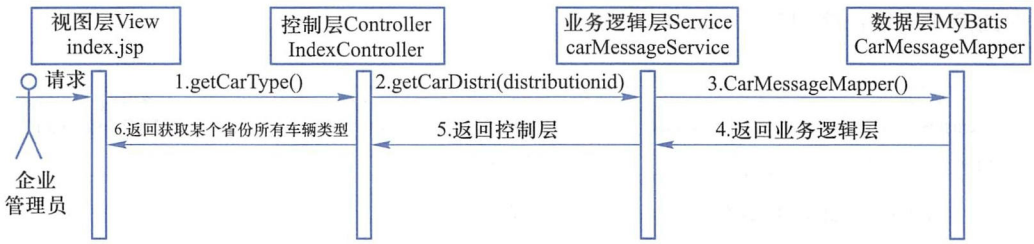


图 4-2-22　不同车型车辆数显示序列图

任务 4.3　设计数据库

【任务描述】

设计并创建新能源汽车智能监控管理系统数据库。

【任务目标】

知识目标

- 了解系统中数据的结构关系。

技能目标

- 能根据数据库设计表在 MySQL 中完成数据库及数据表的创建。

【任务分析】

① 根据系统功能分析，完成数据库的设计。

② 选择 MySQL 数据库管理系统，完成新能源汽车智能监控管理系统的数据库创建。

【任务实现】

1. 数据库设计

① user 表（用户信息表），见表 4-3-1。

表 4-3-1　user 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	varchar（255）	N	PK	会员 ID 标识
2	username	varchar（255）	N		会员账号
3	Password	varchar（255）	N		会员密码
4	createtime	datetime	Y		账号创建时间
5	nickname	varchar（255）	Y		昵称
6	birthdate	date	Y		出生日期
7	sex	int	Y		性别
8	dltime	date	Y		初次拿到驾照时间
9	hpic	varchar（255）	Y		头像图片路径
10	usertype	int	Y		用户类型

② role 表（用户权限类型表），见表 4-3-2。

表 4-3-2　role 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	会员 ID 标识
2	rolename	varchar（255）	N		权限名称
3	roletype	varchar（255）	N		权限值

③ uploadfile 表（文件上传信息表），见表 4-3-3。

表 4-3-3　uploadfile 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	会员 ID 标识
2	filename	varchar（255）	N		上传的文件名
3	filepath	varchar（255）	N		文件路径
4	filetype	varchar（255）	Y		文件类型
5	filetime	datetime	Y		上传时间
10	uploadname	varchar（255）	Y		上传后的文件名

④ carmessage 表（汽车信息表），见表 4-3-4。

表 4-3-4　carmessage 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	vehID	varchar（20）	N		车辆 ID，可以用车架号
3	plateNumber	varchar（20）	Y		车牌

续表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
4	brand	varchar（255）	Y		品牌
5	remarks	varchar（255）	Y		备注
6	producedate	date	Y		出厂日期
7	currentmileage	int	Y		当前里程数
8	userid	varchar（255）	Y		用户 ID
9	createid	varchar（255）	Y		创建人 ID
10	cartype	varchar（255）	Y		车辆类型

⑤ fault 表（故障信息主表），见表 4-3-5。

表 4-3-5　fault 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	faulttype	int	N		故障类型
3	faultnumber	int	Y		故障数量
4	faultname	varchar（255）	Y		故障名称
5	proid	int	Y		参数

⑥ faultinfo 表（故障详情表），见表 4-3-6。

表 4-3-6　faultinfo 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	faultID	varchar（50）	N		故障识别号
3	faultState	varchar（100）	N		故障状态
4	vehID	varchar（50）	N		故障车辆
5	realtime	varchar（100）	N		故障时间
6	faultLevel	varchar（50）	Y		故障级别
7	modID	varchar（50）	Y		故障模块号
8	remarks	varchar（255）	Y		备注

⑦ tbfaultmean 表（故障详情表），见表 4-3-7。

表 4-3-7　tbfaultmean 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	faultID	varchar（50）	N		故障编号
3	faultmean	varchar（50）	Y		故障含义
4	remarks	varchar（255）	Y		备注

⑧ tbfaultsolution 表（故障提醒信息表），见表 4-3-8。

表 4-3-8　tbfaultsolution 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	faultID	varchar（100）	N		故障编号
3	verNumber	int	N		车辆标识
4	faultSolution	varchar（500）	Y		故障提醒
5	remarks	varchar（255）	Y		备注

⑨ distribution 表（省份车辆分布表），见表 4-3-9。

表 4-3-9　distribution 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	carnumber	int	N		车辆数量
3	longtude	decimal	N		经度
4	latitude	decimal	Y		纬度
5	province	varchar（255）	Y		所在省份

⑩ cardistribution 表（车辆分布表），见表 4-3-10。

表 4-3-10　cardistribution 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	carmessageid	int	N		车辆数量
3	distributionid	decimal	N		车辆分布 id

⑪ tbvehmod 表（汽车模块信息表），见表 4-3-11。

表 4-3-11　tbvehmod 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	modID	varchar（50）	N		模块号
3	modNameRe	varchar（50）	Y		模块简称
4	modName	varchar（50）	Y		模块名称
5	remarks	varchar（255）	Y		备注

⑫ tbvehlocation 表（汽车定位信息表），见表 4-3-12。

表 4-3-12　tbvehlocation 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	vehID	varchar（100）	N		车辆 ID，可以用车架号
3	modID	varchar（100）	Y		模块号
4	realtime	varchar（100）	Y		行程时间
5	latitudes	varchar（255）	Y		经度
6	longitudes	varchar（255）	Y		纬度
7	remarks	varchar（255）	Y		城市
8	recordtime	date	Y		记录时间
9	province	varchar（255）	Y		对应的省份

⑬ carfriend 表（车友信息表），见表 4-3-13。

表 4-3-13　carfriend 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	carfname	varchar（255）	N		车友名称
3	cartype	varchar（255）	Y		车辆类型
4	carnumber	varchar（255）	Y		车牌号
5	userid	varchar（255）	Y		用户 id
6	friendid	varchar（255）	Y		车友 id

⑭ Comment 表（车友圈评论表），见表 4-3-14。

表 4-3-14　Comment 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	text	varchar（255）	Y		评论内容
3	contentid	int	Y		评论主题的编号
4	time	datetime	Y		评论时间
5	userid	varchar（255）	Y		评论人

⑮ content 表（车友圈信息表），见表 4-3-15。

表 4-3-15　content 表

序号	字 段 名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	text	varchar（5000）	N		发送的文字信息

续表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
3	praise	int	Y		点赞数
4	comments	int	Y		评论数
5	releasetime	datetime	Y		发布时间
6	carfriendid	int	Y		评论者
7	userid	varchar（255）	Y		发布者

⑯ contentpraise 表（车友圈点赞表），见表 4-3-16。

表 4-3-16　contentpraise 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	contentid	int	Y		点赞的主题编号
3	userid	varchar（255）	Y		点赞的用户编号

⑰ tbvehicles 表（车辆信息表），见表 4-3-17。

表 4-3-17　tbvehicles 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	vehID	varchar（20）	N		车辆 ID，可以用车架号
3	plateNumber	varchar（20）	Y		车牌号
4	remarks	varchar（255）	Y		备注

⑱ tbframe 表（模块参数表），见表 4-3-18。

表 4-3-18　tbframe 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	frameID	varchar（100）	N		帧 ID
3	vehID	varchar（50）	N		车号
4	realtime	varchar（50）	N		数据采集实时时间
5	modID	varchar（50）	Y		模块 ID-发送模块
6	cycle_time	varchar（100）	Y		发送周期
7	para1_name	varchar（100）	Y		参数 1 名
8	para1_value	varchar（100）	Y		参数 1 值
9	Para2_name	varchar（100）	Y		参数 2 名

续表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
10	Para2_value	varchar（100）	Y		参数 2 值
11	Para3_name	varchar（100）	Y		参数 3 名
12	Para3_value	varchar（100）	Y		参数 3 值
13	Para4_name	varchar（100）	Y		参数 4 名
14	Para4_value	varchar（100）	Y		参数 4 值
15	Para5_name	varchar（100）	Y		参数 5 名
16	Para5_value	varchar（100）	Y		参数 5 值
17	Para6_name	varchar（100）	Y		参数 6 名
18	Para6_value	varchar（100）	Y		参数 6 值
19	remarks	varchar（255）	Y		备注

⑲ tbmodpartype 表（模块参数表），见表 4-3-19。

表 4-3-19　tbmodpartype 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	parName	varchar（50）	N		参数名
3	parMean	varchar（100）	N		参数含义
4	parValType	varchar（50）	Y		参数值类型
5	modID	varchar（50）	N		模块 ID-发送模块
6	remarks	varchar（255）	Y		备注

⑳ userrole 表（用户权限表），见表 4-3-20。

表 4-3-20　userrole 表

序号	字　段　名	数 据 类 型	可否为空	主外键	描　　述
1	id	int	N	PK	流水 ID 标识
2	userid	varchar（50）	N		参数名
3	roleid	varchar（100）	N		参数含义
4	status	varchar（50）	Y		参数值类型
5	reply	varchar（50）	N		模块 ID-发送模块

2. 创建数据库和表的 SQL 语句

① 在 MySQL 中新建数据库，名为 energycar，如图 4-3-1 所示。

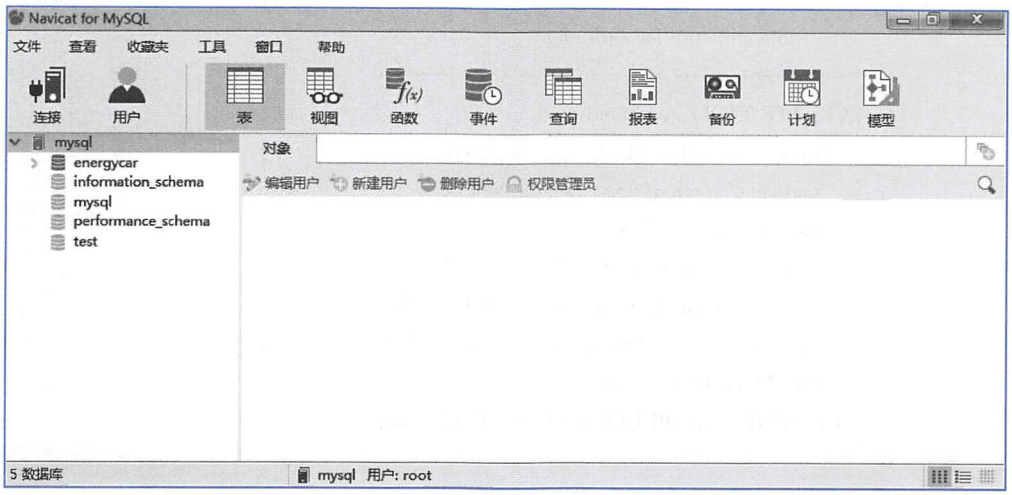


图 4-3-1　新建数据库

② 在新建的数据库执行以下 SQL 语句。

```
<link rel="stylesheet" href="res/bootstrap/css/bootstrap.mi
/*
MySQL Data Transfer
Source Host: 192.168.0.88
Source Database: energycar
Target Host: 192.168.0.88
Target Database: energycar
Date: 2017/7/10 9:35:52
*/
SET FOREIGN_KEY_CHECKS=0;
-- ----------------------------
-- Table structure for c3p0testtable
-- ----------------------------
CREATE TABLE 'c3p0testtable' (
  'a' char(1) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for cardistribution
-- ----------------------------
CREATE TABLE 'cardistribution' (
  'id' int(11) NOT NULL auto_increment COMMENT '主键',
  'carmessageid' int(11) default NULL COMMENT '车辆 id',
  'distributionid' int(11) default NULL COMMENT '车辆分布 id',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- ----------------------------
```

```
-- Table structure for carfriend
-- ----------------------------
CREATE TABLE 'carfriend' (
  'id' int(11) NOT NULL auto_increment,
  'carfname' varchar(255) default NULL,
  'cartype' varchar(255) default NULL,
  'carnumber' varchar(255) default NULL,
  'userid' varchar(255) default NULL COMMENT '用户 id',
  'friendid' varchar(255) default NULL COMMENT '车友 id',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for carmessage
-- ----------------------------
CREATE TABLE 'carmessage' (
  'id' int(11) NOT NULL auto_increment COMMENT '流水 ID',
  'vehID' varchar(20) NOT NULL COMMENT '车辆 ID,可以用车架号',
  'plateNumber' varchar(50) default NULL COMMENT '车牌',
  'brand' varchar(255) default NULL COMMENT '品牌',
  'remarks' varchar(255) default NULL COMMENT '备注',
  'producedate' date default NULL COMMENT '出厂日期',
  'currentmileage' int(255) default NULL COMMENT '当前里程数',
  'userid' varchar(255) default NULL COMMENT '用户 id',
  'createid' varchar(255) default NULL COMMENT '创建人 id',
  'cartype' varchar(255) default NULL COMMENT '车辆类型',
  PRIMARY KEY  ('id'),
  UNIQUE KEY 'user_id' ('userid')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for comment
-- ----------------------------
CREATE TABLE 'comment' (
  'id' int(11) NOT NULL auto_increment COMMENT '主键',
  'text' varchar(255) default NULL COMMENT '评论内容',
  'contentid' int(11) default NULL,
  'time' datetime default NULL COMMENT '评论时间',
  'userid' varchar(255) default NULL COMMENT '评论人',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for content
-- ----------------------------
CREATE TABLE 'content' (
  'id' int(255) NOT NULL auto_increment COMMENT '主键 id',
  'txt' varchar(5000) default NULL COMMENT '发送的文字信息',
  'praise' int(255) default NULL COMMENT '点赞',
  'comments' int(255) default NULL COMMENT '评论',
  'releasetime' datetime default NULL COMMENT '发布时间',
  'carfriendid' int(11) default NULL,
  'userid' varchar(255) default NULL,
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for contentpraise
-- ----------------------------
CREATE TABLE 'contentpraise' (
  'id' int(11) NOT NULL auto_increment,
  'contentid' int(11) default NULL,
  'userid' varchar(255) default NULL,
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for distribution
-- ----------------------------
CREATE TABLE 'distribution' (
  'id' int(11) NOT NULL auto_increment COMMENT '主键',
  'carnumber' int(11) default NULL COMMENT '车辆数量',
  'longtude' decimal(10,7) default NULL COMMENT '经度',
  'latitude' decimal(10,7) default NULL COMMENT '纬度',
  'province' varchar(255) default NULL COMMENT '所在省份',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for fault
-- ----------------------------
CREATE TABLE 'fault' (
  'id' int(11) NOT NULL auto_increment COMMENT '主键',
  'faulttype' int(11) default NULL COMMENT '故障类型',
  'faultnumber' int(11) default NULL COMMENT '故障数量',
```

```
  'faultname' varchar(255) default NULL,
  'proid' int(11) default NULL,
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for faultinfo
-- ----------------------------
CREATE TABLE 'faultinfo' (
  'id' int(11) NOT NULL,
  'faultID' varchar(50) NOT NULL,
  'faultState' varchar(100) NOT NULL,
  'vehID' varchar(50) NOT NULL,
  'realtime' varchar(100) NOT NULL,
  'faultLevel' varchar(50) default NULL,
  'modID' varchar(50) default NULL,
  'remarks' varchar(255) default NULL,
  PRIMARY KEY  ('id'),
  UNIQUE KEY 'realtime' ('realtime'),
  KEY 'modID' ('modID')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for role
-- ----------------------------
CREATE TABLE 'role' (
  'id' int(11) NOT NULL auto_increment,
  'rolename' varchar(255) default NULL,
  'roletype' varchar(255) default NULL,
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for tbfaultmean
-- ----------------------------
CREATE TABLE 'tbfaultmean' (
  'Id' int(11) NOT NULL auto_increment,
  'faultID' varchar(50) NOT NULL,
  'faultmean' varchar(50) default NULL,
  'remarks' varchar(255) default NULL,
  PRIMARY KEY  ('Id'),
  KEY 'faultID' ('faultID')
```

```
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for tbfaultsolution
-- ----------------------------
CREATE TABLE 'tbfaultsolution' (
  'id' int(11) NOT NULL auto_increment,
  'faultID' varchar(100) NOT NULL,
  'verNumber' int(2) NOT NULL,
  'faultSolution' varchar(500) default NULL,
  'remarks' varchar(255) default NULL,
  PRIMARY KEY  ('id','faultID','verNumber'),
  UNIQUE KEY 'faultID' ('faultID')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for tbframe
-- ----------------------------
CREATE TABLE 'tbframe' (
  'Id' int(11) NOT NULL auto_increment COMMENT '流水 id',
  'frameID' varchar(100) NOT NULL COMMENT '帧 id',
  'vehID' varchar(50) NOT NULL COMMENT '车号',
  'realtime' varchar(50) NOT NULL COMMENT '数据采集实时时间',
  'modID' varchar(50) NOT NULL COMMENT '模块 ID-发送模块',
  'cycle_time' varchar(100) default NULL COMMENT '发送周期',
  'para1_name' varchar(100) default NULL COMMENT '参数 1 名',
  'para1_value' varchar(100) default NULL COMMENT '参数 1 值',
  'para2_name' varchar(100) default NULL COMMENT '参数 2 名',
  'para2_value' varchar(100) default NULL COMMENT '参数 2 值',
  'para3_name' varchar(100) default NULL COMMENT '参数 3 名',
  'para3_value' varchar(100) default NULL COMMENT '参数 3 值',
  'para4_name' varchar(100) default NULL COMMENT '参数 4 名',
  'para4_value' varchar(100) default NULL COMMENT '参数 4 值',
  'para5_name' varchar(100) default NULL COMMENT '参数 5 名',
  'para5_value' varchar(100) default NULL COMMENT '参数 5 值',
  'para6_name' varchar(100) default NULL COMMENT '参数 6 名',
  'para6_value' varchar(100) default NULL COMMENT '参数 6 值',
  'remarks' varchar(255) default NULL COMMENT '备注',
  PRIMARY KEY  ('Id'),
  KEY 'modID' ('modID')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for tbmodpartype
-- ----------------------------
CREATE TABLE 'tbmodpartype' (
  'Id' int(11) NOT NULL auto_increment COMMENT '流水 id',
  'parName' varchar(50) NOT NULL COMMENT '参数名',
  'parMean' varchar(100) default NULL COMMENT '参数含义',
  'parValType' varchar(50) default NULL COMMENT '参数值类型',
  'modID' varchar(50) NOT NULL COMMENT '模块 id',
  'remarks' varchar(255) default NULL COMMENT '备注',
  PRIMARY KEY  ('Id'),
  UNIQUE KEY 'parName' ('parName'),
  KEY 'modID' ('modID')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for tbvehicles
-- ----------------------------
CREATE TABLE 'tbvehicles' (
  'Id' int(11) NOT NULL auto_increment,
  'vehID' varchar(20) NOT NULL,
  'plateNumber' varchar(20) default NULL,
  'remarks' varchar(255) default NULL,
  PRIMARY KEY  ('Id')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for tbvehlocation
-- ----------------------------
CREATE TABLE 'tbvehlocation' (
  'id' int(11) NOT NULL auto_increment,
  'vehID' varchar(100) default NULL,
  'modID' varchar(100) default NULL,
  'realtime' varchar(100) default NULL,
  'latitudes' varchar(255) default NULL,
  'longitudes' varchar(255) default NULL,
  'remarks' varchar(255) default NULL,
  'recordtime' date default NULL,
  'province' varchar(255) default NULL COMMENT '对应的省份',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for tbvehmod
-- ----------------------------
CREATE TABLE 'tbvehmod' (
  'id' int(11) NOT NULL auto_increment,
  'modID' varchar(50) NOT NULL,
  'modNameRe' varchar(50) default NULL,
  'modName' varchar(50) default NULL,
  'remarks' varchar(255) default NULL,
  PRIMARY KEY  ('modID','id')
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for uploadfile
-- ----------------------------
CREATE TABLE 'uploadfile' (
  'id' int(11) NOT NULL auto_increment COMMENT '主键',
  'filename' varchar(255) default NULL COMMENT '上传的文件名',
  'filepath' varchar(255) default NULL COMMENT '文件路径',
  'filetype' varchar(255) default NULL COMMENT '文件类型',
  'filetime' datetime default NULL COMMENT '上传时间',
  'uploadname' varchar(255) default NULL COMMENT '上传后的文件名',
  PRIMARY KEY  ('id')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

-- ----------------------------
-- Table structure for user
-- ----------------------------
CREATE TABLE 'user' (
  'id' varchar(255) NOT NULL COMMENT '主键',
  'username' varchar(255) default NULL,
  'password' varchar(255) default NULL,
  'createtime' datetime default NULL COMMENT '账号创建时间',
  'nickname' varchar(255) default NULL COMMENT '昵称',
  'birthdate' date default NULL COMMENT '出生日期',
  'sex' int(4) default NULL COMMENT '性别',
  'dltime' date default NULL COMMENT '初次拿到驾照时间',
  'hpic' varchar(255) default NULL COMMENT '头像图片路径',
  'usertype' int(255) default NULL COMMENT '用户类型',
  PRIMARY KEY  ('id'),
  UNIQUE KEY 'username' ('username')
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
-- ----------------------------
-- Table structure for userrole
-- ----------------------------
CREATE TABLE 'userrole' (
  'id' int(11) NOT NULL auto_increment,
  'userid' varchar(255) default NULL,
  'roleid' int(11) default NULL,
  'status' int(11) default NULL,
  'reply' varchar(255) default NULL,
  PRIMARY KEY  ('id'),
  KEY 'userrole_1' ('roleid'),
  KEY 'userrole_2' ('userid'),
  CONSTRAINT 'userrole_2' FOREIGN KEY ('userid') REFERENCES 'user' ('id') ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT 'userrole_1' FOREIGN KEY ('roleid') REFERENCES 'role' ('id') ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

技能训练

车友圈详细设计。展现车友圈模块功能的用例图和类图。

任务描述

对车友圈模块分析，并按要求绘制相关 UML 模型图。

任务分析

车友圈类似于微信朋友圈，车主可以添加车友，也可以发布友圈信息，同时可对车友圈发布的内容进行点赞和评论，提供了一个车友交流平台。

项目总结

需求分析是软件开发的重要阶段，本项目重点对教学项目“新能源汽车智能监控管理系统”进行了需求分析，详细介绍了项目的背景知识，对系统进行了功能分析、原型设计、概要设计，并绘制了标准的建模图，为项目的实施做了准备。

项目 5　构建本地开发环境

PPT 构建本地开发环境

项目描述

搭建开发环境是 Java Web 开发的第一步，本项目将带领读者一起安装和配置 Java Web 应用程序的开发环境，包括 Java 开发包 JDK（Java Development Kit）的安装，应用服务器软件 Tomcat、MySQL、集成开发环境 Eclipse 的安装及配置。 在本项目中，读者还将在搭建好的开发环境中创建第一个 Java Web 工程，并进行代码的编写、运行和调试，初步了解 Java Web 应用程序的框架和一些关于 Java Web 开发的基础知识。

知识目标

- 了解 Java Web 开发环境。
- 掌握 JDK 的下载、安装与配置方法。
- 掌握 Tomcat 的下载、安装与配置方法。
- 掌握 Eclipse 集成开发环境的安装与配置方法。
- 掌握 MySQL 数据库管理软件的安装方法。
- 了解 Java Web 项目的基本结构。
- 掌握在 MyEclipse 中创建、发布、运行 Java Web 项目的方法。

技能目标

- 学会安装并配置 JDK、Tomcat、MySQL、Eclipse，搭建 Java Web 开发环境。
- 能使用 Eclipse 创建、发布并运行 Java Web 项目。

任务列表

任务编号	任务名称	建议课时
任务 5.1	安装与配置 JDK	0.25
任务 5.2	安装与配置 Tomcat	0.25
任务 5.3	安装与配置 Eclipse	0.25
任务 5.4	创建 Java Web 项目	0.5
任务 5.5	安装 MySQL 数据库	0.25
技能训练	搭建开发环境	0.5
	总计：	2 课时

任务 5.1　安装与配置 JDK

微课 5-1
任务 5.1

【任务描述】

JDK（Java Development Kit，Java 开发包）是整个 Java 的核心，包括了 Java 运行环境、Java 工具和 Java 基础类库。JDK 作为 Java 开发的环境，不管是做 Java 开发还是做安卓开发，都必须在计算机上安装 JDK。在本任务中，将完成 JDK 的安装与配置，如图 5-1-1 所示。

```
C:\Users\zhud>javac
用法: javac <options> <source files>
其中, 可能的选项包括:
  -g                         生成所有调试信息
  -g:none                    不生成任何调试信息
  -g:{lines,vars,source}     只生成某些调试信息
  -nowarn                    不生成任何警告
  -verbose                   输出有关编译器正在执行的操作的消息
  -deprecation               输出使用已过时的 API 的源位置
  -classpath <路径>            指定查找用户类文件和注释处理程序的位置
  -cp <路径>                   指定查找用户类文件和注释处理程序的位置
  -sourcepath <路径>           指定查找输入源文件的位置
  -bootclasspath <路径>        覆盖引导类文件的位置
  -extdirs <目录>              覆盖所安装扩展的位置
  -endorseddirs <目录>         覆盖签名的标准路径的位置
  -proc:{none,only}          控制是否执行注释处理和/或编译。
  -processor <class1>[,<class2>,<class3>...] 要运行的注释处理程序的名称; 绕过默
认的搜索进程
  -processorpath <路径>        指定查找注释处理程序的位置
  -d <目录>                    指定放置生成的类文件的位置
  -s <目录>                    指定放置生成的源文件的位置
  -implicit:{none,class}     指定是否为隐式引用文件生成类文件
  -encoding <编码>             指定源文件使用的字符编码
  -source <发行版>             提供与指定发行版的源兼容性
  -target <发行版>             生成特定 VM 版本的类文件
  -version                   版本信息
  -help                      输出标准选项的提要
  -A关键字[=值]                传递给注释处理程序的选项
  -X                         输出非标准选项的提要
  -J<标记>                     直接将 <标记> 传递给运行时系统
  -Werror                    出现警告时终止编译
  @<文件名>                    从文件读取选项和文件名
```

图 5-1-1　JDK 运行界面

【任务目标】

知识目标

- 了解 JDK 的作用。
- 掌握 JDK 的下载、安装与配置方法。

技能目标

- 能够正确安装与配置 JDK。

【任务分析】

安装 JDK 是搭建 Java Web 开发环境的第一步。首先要准备好 JDK 的安装文件，接着根据安装向导的提示进行安装。JDK 安装完毕后，还需要对其进行配置，最后通过测试，检验 JDK 的安装与配置的正确性。

【任务实施】

1. 下载 JDK 安装文件

如果没有 JDK 安装文件，可以在甲骨文公司的官方网站上下载，下载界面如图 5-1-2 所示，下载后的安装文件如图 5-1-3 所示。本书中介绍使用的是 JDK 7，读者可以根据实际情况自行选择。

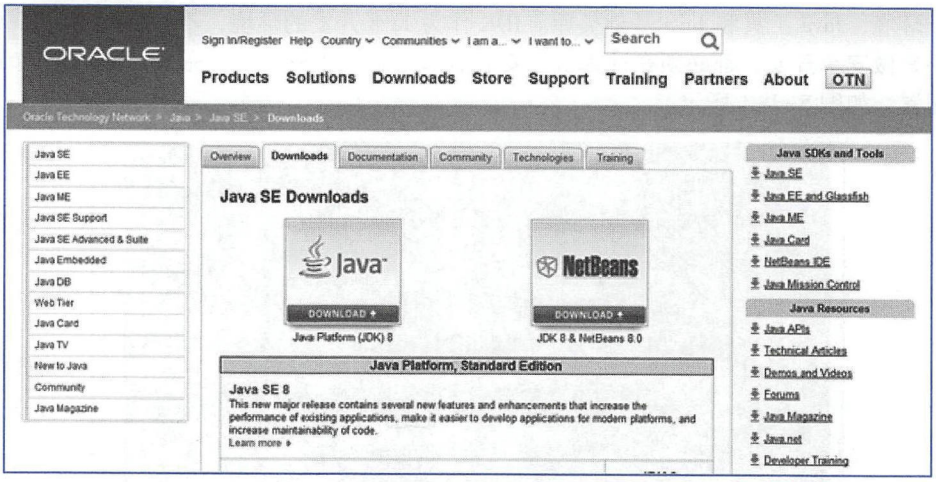


图 5-1-2　JDK 下载界面

2. 安装 JDK

① 双击运行已载的 JDK 安装文件，将出现如图 5-1-4 所示的安装欢迎界面，接下来只要按照安装向导的提示进行操作即可。

图 5-1-3　JDK 7 安装文件

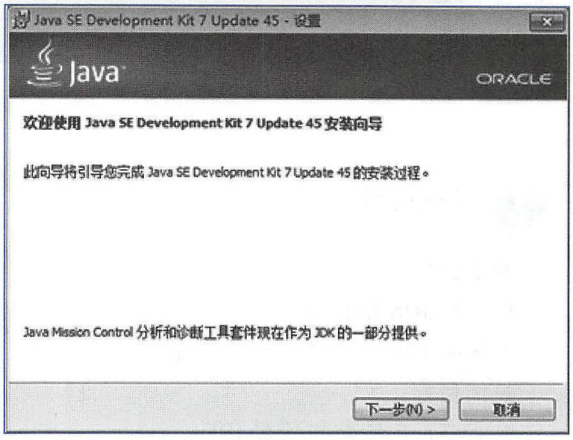
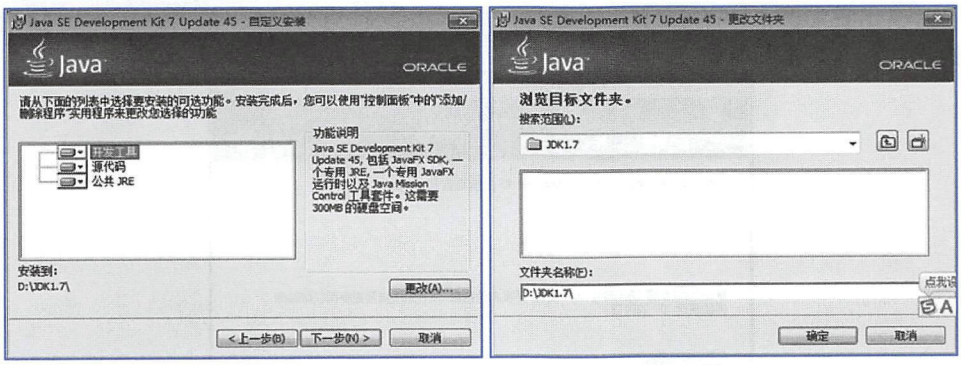


图 5-1-4　JDK 安装欢迎界面

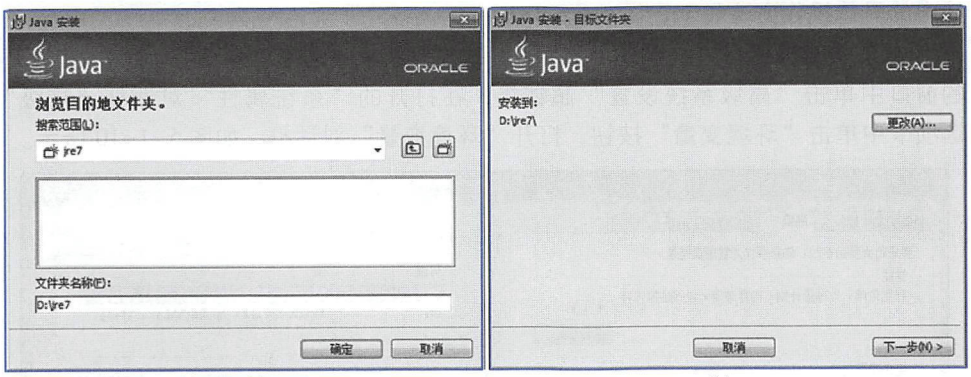
② 在如图 5-1-5 所示的界面中，用户可以根据需要选择安装路径和安装内容。注意，在设置 JDK 安装路径时，建议放在 C:\JDK1.7 或 D:\JDK1.7 有空格字符的目录文件夹下，避免在以后编译、运行时因文件路径而出错。这里安装到 D:\JDK1.7 目录下。



(a)　　(b)

图 5-1-5　选择 JDK 安装路径

③ JDK 安装快结束时，会出现安装 JRE 的对话框，同安装 JDK 一样，用户也可以自行设置 JRE 的安装路径进行安装，如图 5-1-6 所示。建议如图 5-1-7 所示将 JDK 和 JRE 都安装在同一个文件夹中的不同子文件夹中（不要都安装在同一个文件夹的根目录下，否则会出错）。最后单击“下一步”按钮进行安装，如图 5-1-8 所示。



(a)　　(b)

图 5-1-6　选择 JRE 安装路径

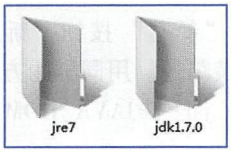


图 5-1-7　JDK 和 JRE 安装在不同的文件夹中

图 5-1-8　正在安装 Java 界面

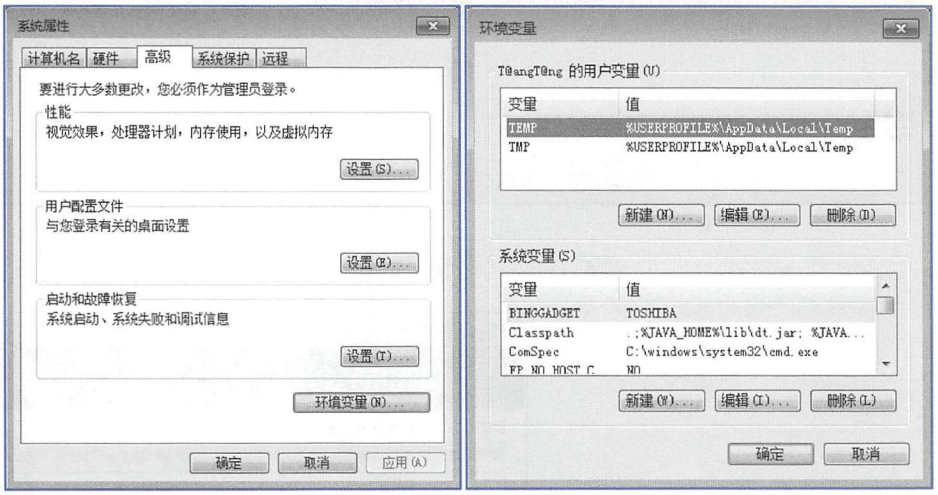
当出现如图 5-1-9 所示的界面时，JDK 的安装已经完成，单击“关闭”按钮退出。

图 5-1-9　Java 安装成功界面

3. 配置 JDK 环境变量

JDK 安装完成后，需要配置环境变量才能使用 JDK 开发，在 Windows 7 操作系统中的具体操作方法如下：

① 右击桌面图标“计算机”，在弹出的快捷菜单中选择“属性”命令，在打开的窗口中单击“高级系统设置”超链接，在打开的“系统属性”对话框“高级”选项卡中单击“环境变量”按钮，打开“环境变量”对话框，如图 5-1-10 所示。

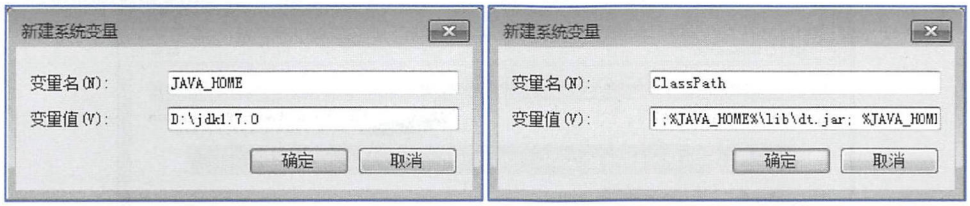


(a)　　(b)

图 5-1-10　环境变量设置界面

② 单击“环境变量”对话框“系统变量”框下的“新建”按钮，新建名为“JAVA_HOME”的系统变量，变量的值为 JDK 的实际安装路径。用同样的方法，再新建名为“ClassPath”，值为“.;%JAVA_HOME%\lib\dt.jar; %JAVA_HOME%\lib\tools.jar”的系统变量，如图 5-1-11 所示。

③ 在“系统变量”中双击 Path 变量，在打开的“编辑系统变量”对话框中为 Path 变量添加值“%JAVA_HOME%\bin;”，如图 5-1-12 所示。



(a)　　(b)

图 5-1-11　“新建系统变量”对话框

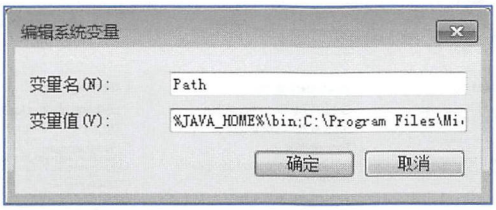


图 5-1-12　编辑系统变量

4. 测试配置是否成功

设置好 JDK 的环境变量后，在“开始”菜单中选择“附件→命令提示符”命令，在打开的窗口光标后输入“java-version”命令，若出现如图 5-1-13 所示，显示版本信息，则说明 JDK 安装和配置成功，系统环境变量被更新，下面就可以进行 Web 服务器的安装配置了。如果显示的内容与图 5-1-13 不同，则需要重新安装。

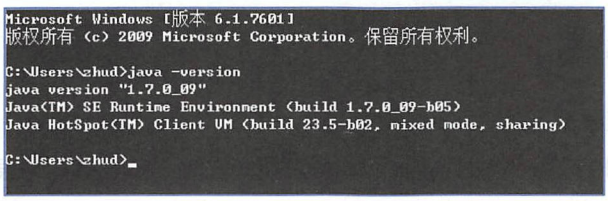


图 5-1-13　JDK 配置测试界面

任务 5.2　安装与配置 Tomcat

微课 5-2
任务 5.2

【任务描述】

Tomcat 是 Java Web 项目运行的常用应用服务器软件之一。在本任务中，将完成 Tomcat 的下载、部署与测试，成功部署 Tomcat 的效果如图 5-2-1 所示。

【任务目标】

知识目标

- 了解 Web 服务器的作用及常用的 Web 服务器。
- 掌握 Tomcat 的下载、部署与测试方法。

技能目标

- 能够正确部署 Tomcat。

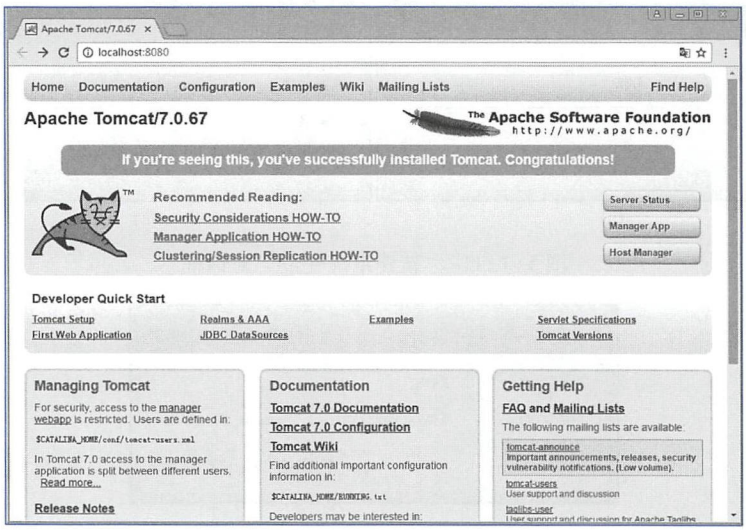


图 5-2-1　Tomcat 成功部署界面

【任务分析】

Tomcat 服务器是一个免费开放源代码的 Web 应用服务器。安装 Tomcat 服务器软件，先要下载 Tomcat 安装文件，解压缩安装文件，启动 Tomcat，通过测试检验 Tomcat 是否安装成功。本书将使用 Tomcat 7.0 进行安装。

【任务实施】

1. 下载安装 Tomcat

① 从 Apache 官网下载 Tomcat，本书采用 Tomcat 7.0 开发。在导航栏中，单击“Tomcat 7”按钮进入，在下载界面根据当前计算机的操作系统选择对应版本进行下载，如图 5-2-2 所示。

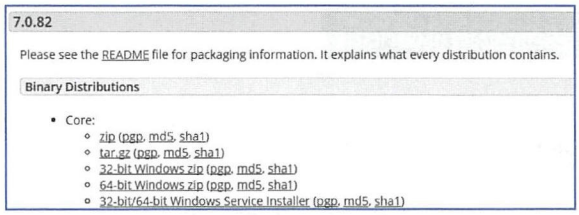


图 5-2-2　Tomcat 下载界面

前 4 项都是可以直接解压缩使用的版本，最后一项 32-bit/64-bit Windows Service Installer 则是需要在计算机安装的版本。根据操作系统选择合适版本，如当前计算机是 Windows 64 位的操作系统，那么需要下载 64-bit windows zip 版本。

② 将下载好的 Tomcat 压缩包解压至本地磁盘，如图 5-2-3 所示。

注意

Tomcat 的存储路径最好不要包含有中文。

图 5-2-3　Tomcat 目录结构

2. 运行测试 Tomcat

① 在 Tomcat 文件夹下进入 bin 子文件夹，双击 startup. bat 文件，启动 Tomcat 服务器，如图 5-2-4 所示。

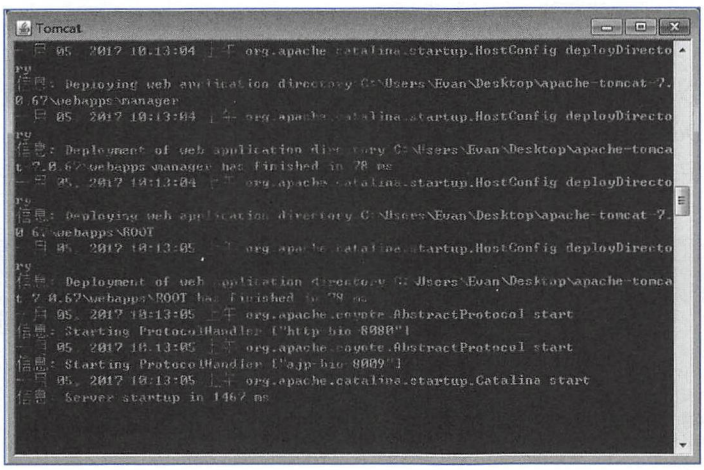


图 5-2-4　启动 Tomcat 服务器

② 在浏览器的地址栏文本框中输入 http://localhost:8080，显示如图 5-2-1 所示界面，则表示 Tomcat 服务器部署成功。

任务 5.3　安装与配置 Eclipse

【任务描述】

微课 5-3
任务 5.3

Eclipse 是功能丰富的 JavaEE 集成开发环境。在本任务中，学生将完成 Java

Web 应用程序的集成开发环境（IDE）Eclipse 的安装与配置。如图 5-3-1 所示是 Eclipse 软件启动时的欢迎界面。本书采用的是 Eclipse LUNA 版本。

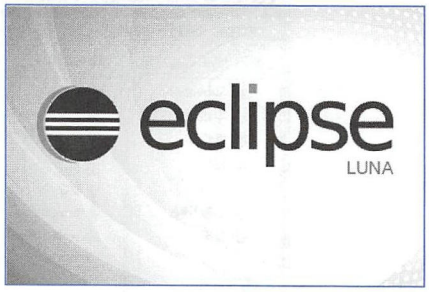


图 5-3-1　Eclipse 启动界面

【任务目标】

知识目标

- 掌握 Eclipse 下载与部署的方法。
- 掌握在 Eclipse 中配置 Tomcat 服务器的方法。

技能目标

- 能正确安装 Eclipse。
- 会在 Eclipse 中正确配置 Tomcat 服务器。

【任务分析】

Eclipse 是一款开源免费软件，从 Eclipse 官网可以很方便地获得，通过下载、安装、配置等步骤完成本任务，在 Eclipse 中必须正确配置 Tomcat，才能保证 Web 项目的部署运行。本书将带领读者一起进行 Eclipse luna 的安装与配置。

【任务实施】

1. 下载安装 Eclipse

读者可以从 Eclipse 的官网获取到各种版本的 Eclipse，一般选择下载 Eclipse IDE for Java EE Developers。

将下载好的 Eclipse 压缩包解压至计算机的本地磁盘，解压好的文件如图 5-3-2 所示。

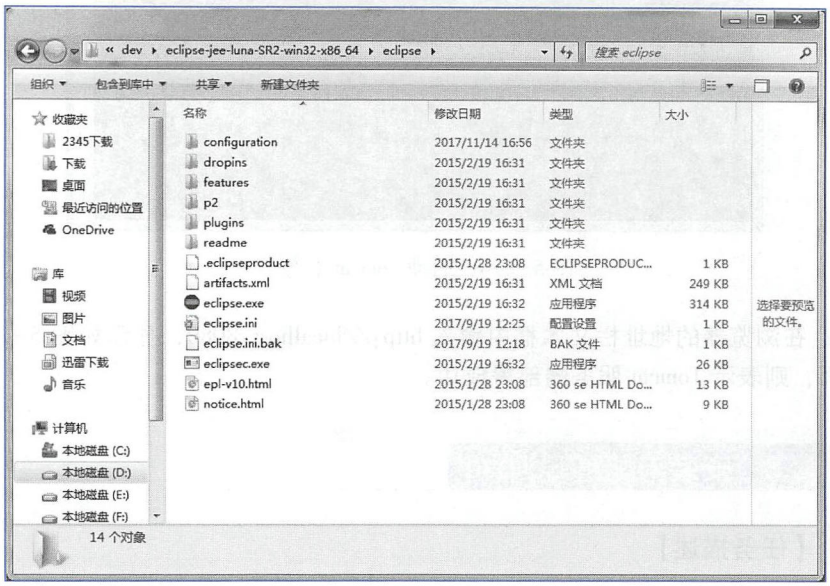


图 5-3-2　Eclipse 目录结构

2. 运行配置 Eclipse

① 双击 eclipse. exe 文件启动 Eclipse，选择工作目录，该目录不能有中文信息，单击“OK”按钮，如图 5-3-3 所示。

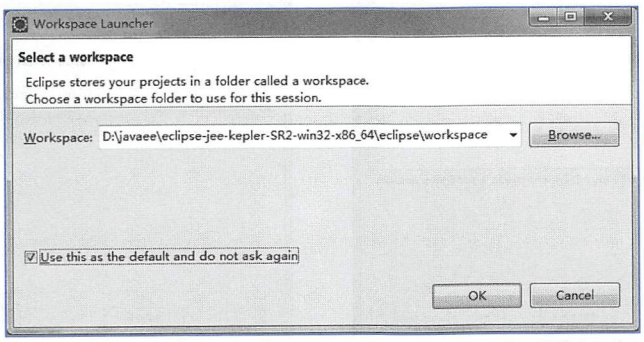


图 5-3-3　Eclipse 选择工作目录

② 配置 Eclipse 的 Web 服务器。在主菜单栏中选择“Window→Preferences”菜单命令，在打开的窗口左侧选择“Server→Runtime Environments”选项，如图 5-3-4 所示。

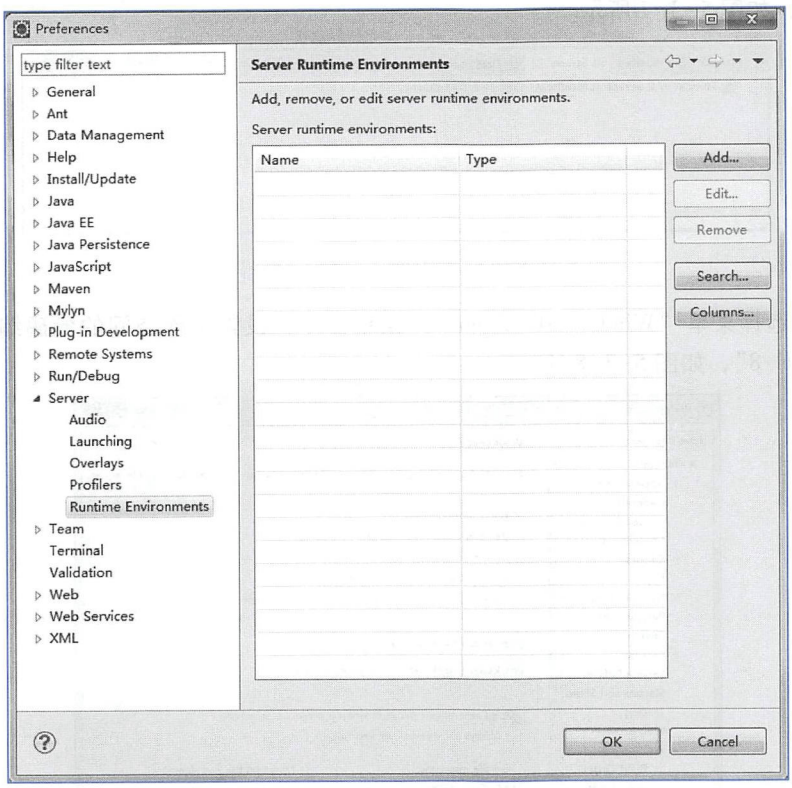


图 5-3-4　配置服务运行环境

③ 单击右侧“Add”按钮添加 Tomcat 服务器，如图 5-3-5 所示。

④ 选择“Apache Tomcat v7. 0”选项，单击“Next”按钮，设置 Tomcat 安装路径，最后单击“Finish”按钮完成配置，如图 5-3-6 所示。

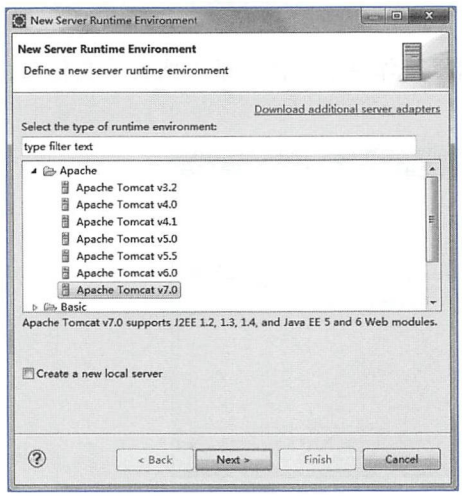


图 5-3-5　选择 Tomcat 服务器

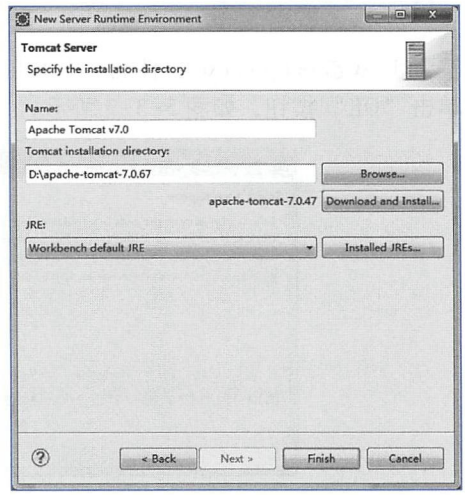


图 5-3-6　配置 Tomcat

⑤ 在主菜单栏中选择“Window→Show View→Servers”菜单命令，打开 Servers 窗口，添加刚才创建的 Tomcat 服务器，在此窗口可以直接停止或者启动调试 Tomcat 服务器，如图 5-3-7 所示。

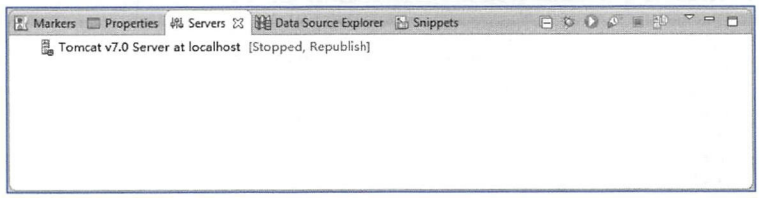


图 5-3-7　添加 Tomcat 服务器

⑥ 选择菜单“Window→Preference”菜单命令，设置工作空间的代码编码方式为“UTF-8”，如图 5-3-8 所示。

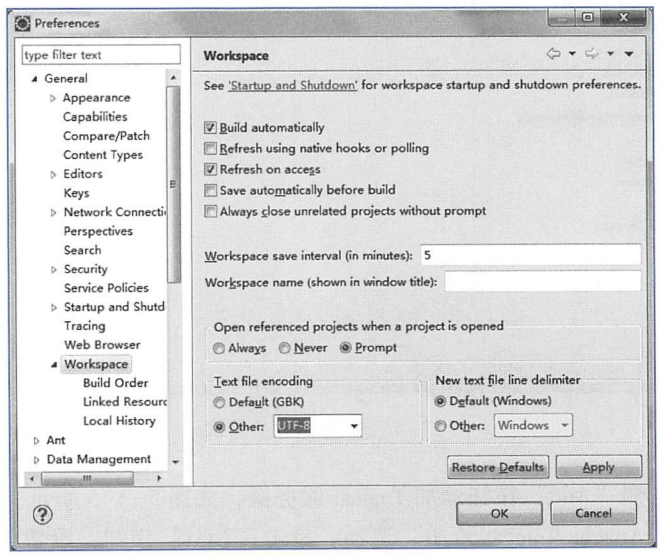


图 5-3-8　设置程序编码方式

任务 5.4　创建 Java Web 项目

微课 5-4
任务 5.4

【任务描述】

在本任务中，学生将在 Eclipse 集成开发环境中创建第一个 Java Web 项目，了解 Web 项目的基本结构，学习如何在 Eclipse 中部署并运行项目。

【任务目标】

知识目标

- 了解 Java Web 项目的基本结构。
- 熟悉 Eclipse 开发环境的界面构成。
- 掌握如何在 Eclipse 中创建、发布、运行 Web 项目。

技能目标

- 能够在 Eclipse 中正确创建 Web 项目。
- 能够在 Eclipse 中正确发布及运行 Web 项目。

【任务分析】

使用 Eclipse 进行 Java Web 开发，首先要创建一个 Dynamic Web Project 项目，在项目中编写程序代码，在 Tomcat 中成功发布后，就可以在浏览器中进行 Web 页面的浏览了。

【任务实施】

1. 新建 HelloWorld 工程

① 选择 Eclipse 主菜单的“File→New→Dynamic Web Project”菜单命令，创建一个 Web 工程，如图 5-4-1 所示。

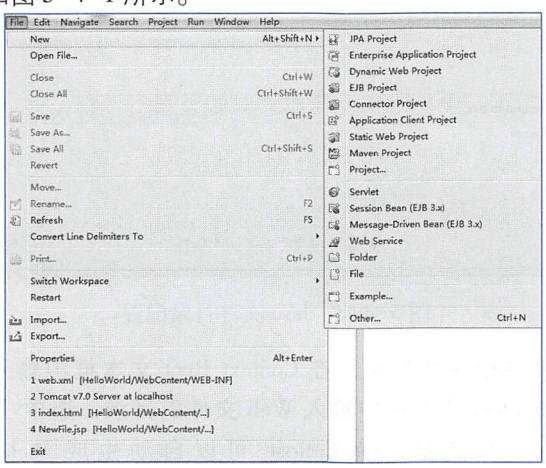


图 5-4-1　新建 Web 工程

② 输入项目名称 HelloWorld，选择之前配置好的 Tomcat 7.0 服务器，单击“Next”按钮，如图 5-4-2 所示。

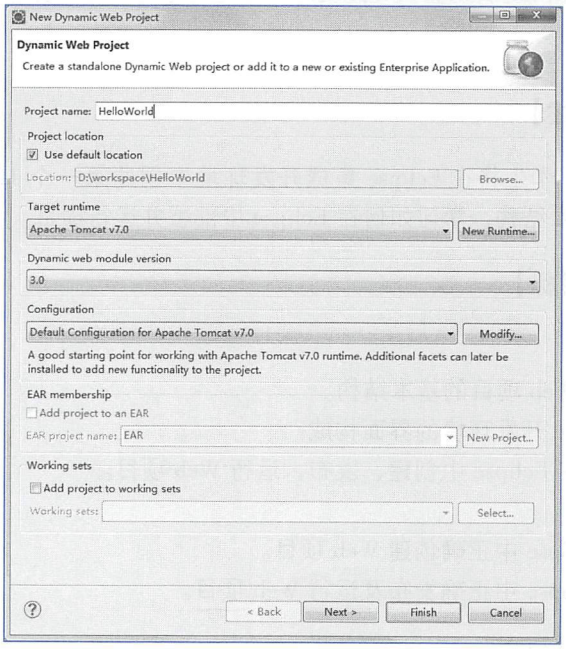


图 5-4-2　Web 工程配置

③ 设置 Web 项目存放 Java 源代码的目录，默认是 src 目录，单击“Next”按钮，如图 5-4-3 所示。

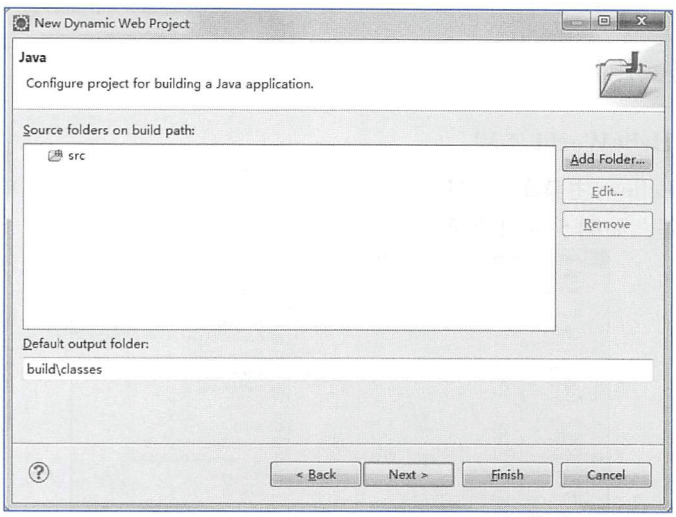


图 5-4-3　Java 文件目录配置

④ 设置 Web 文件相关的目录，在 Context Root 文本框中输入工程的名称 HelloWorld，Content Directory 文本框中输入 Web 文件的目录，选中“Generate Web. xml delopment descriptor”复选框，使 Eclipse 可以自动生成 Web. xml 文件。单击“Finish”按钮完成项目工程的创建，如图 5-4-4 所示。

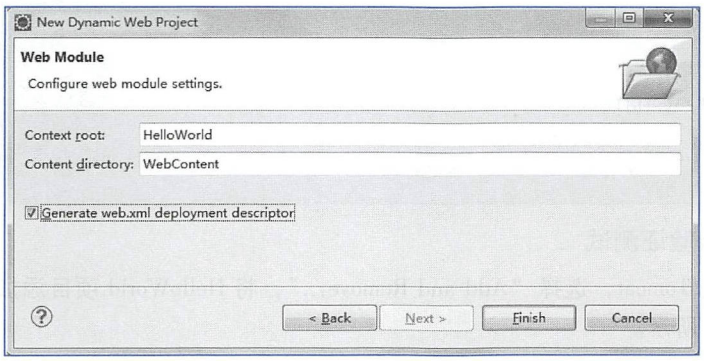


图 5-4-4　Web 文件目录配置

2. 新建 Web 页面

Web 项目新建完成之后，还需要给项目新建一个可以访问的 Web 页面，命名为 index. html。

① 新建完成的 Web 项目结构，如图 5-4-5 所示。

② 右击项目名，在弹出的菜单中选择“New”菜单命令，在 WebContent 目录下创建一个 HTML File 文件，并命名为 index. html，如图 5-4-6 所示。

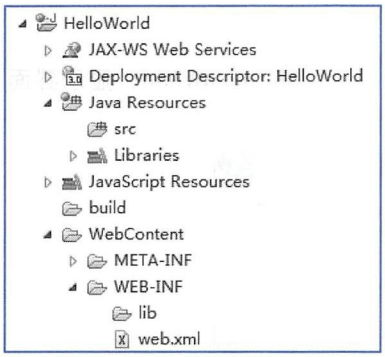


图 5-4-5　Web 项目的结构

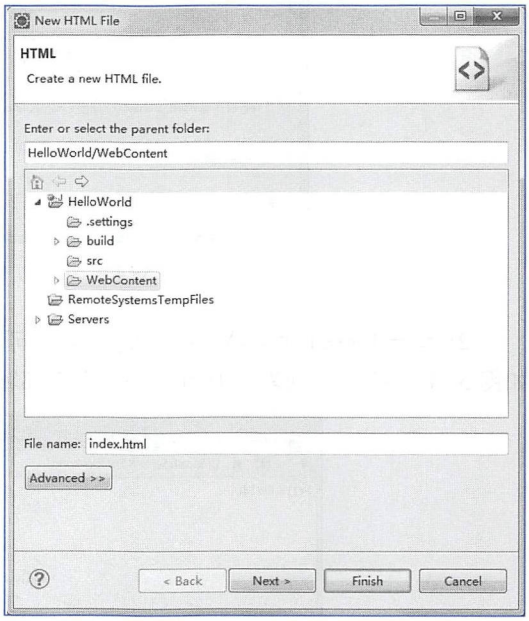


图 5-4-6　新建 HTML 文件

③ 编辑 index. html 文件，输入如下代码。

```
<!DOCTYPEhtml>
<html>
<head>
<meta charset="UTF-8">
<title>My Web Project</title>
```

```
</head>
<body>
    Hello World!
</body>
</html>
```

3. 功能验证测试

① 右击 Tomcat，选择“Add and Remove...”，将 HelloWorld 项目添加至 Tomcat，如图 5-4-7 所示。

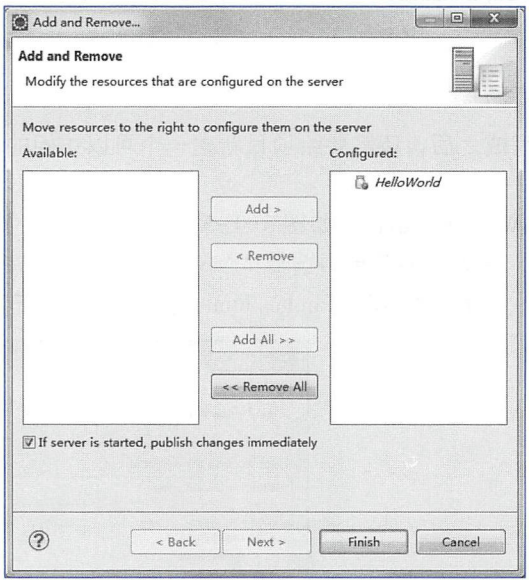


图 5-4-7　添加 Web 项目

② 运行 Tomcat 服务器。打开浏览器访问 localhost：8080/HelloWorld，显示界面如图 5-4-8 所示，则表示 HelloWorld 程序部署成功。

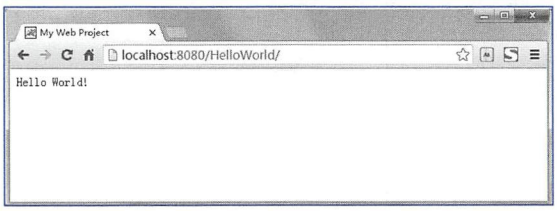


图 5-4-8　HelloWord 项目发布

微课 5-5
任务 5.5

任务 5.5　安装 MySQL 数据库

【任务描述】

在本任务中，将完成 MySQL 数据库服务器与其图形化客户端 Navicat 软件的下载、安装与配置。相关界面如图 5-5-1 和图 5-5-2 所示。

```
Enter password: ******
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.5.30 MySQL Community Server (GPL)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

图 5-5-1　MySQL 数据库服务器界面

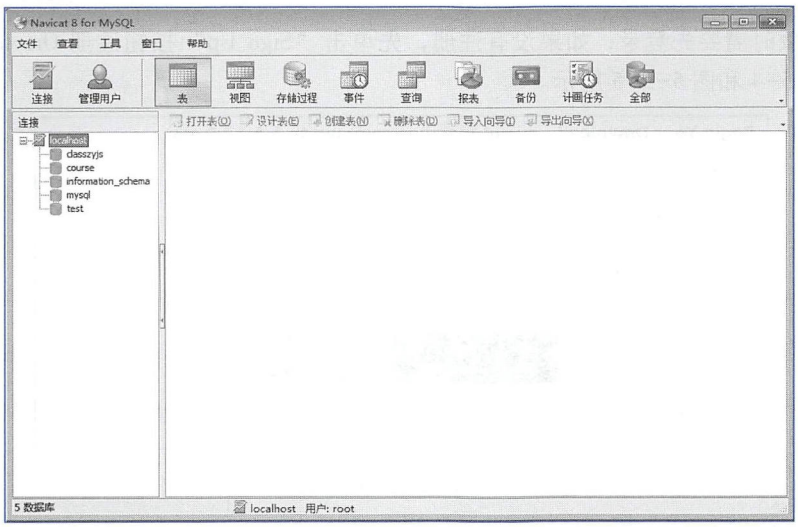


图 5-5-2　Navicat 界面

【任务目标】

知识目标

- 了解 MySQL 数据库的安装过程和方法。

技能目标

- 能够正确下载、安装并配置 MySQL 数据库服务器和图形化客户端软件 Navicat。

【任务分析】

MySQL 数据库软件是一款开源的关系型数据库管理系统，可以从其官网免费获取，但是 MySQL 本身是没有图形用户界面的，因此为了方便起见，需要安装一款 MySQL 的图形界面软件 Navicat。

【任务实施】

1. 下载 MySQL 安装文件

目前最新的 MySQL 版本为 MySQL 5.6.17，可以从官网下载该软件，如图 5-5-3 所示。

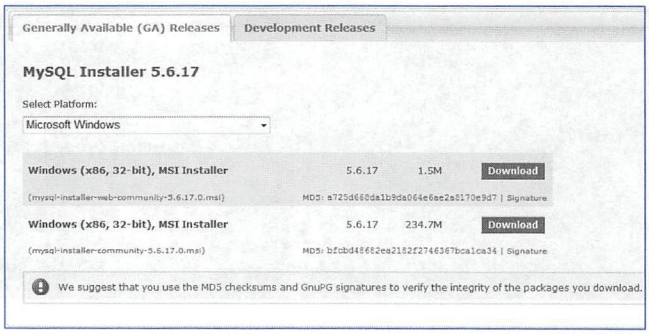


图 5-5-3　下载 MySQL 5.6.17

下载时需要先登录，如果没有注册，先单击“Sign Up”按钮注册 Oracle 账号，如图 5-5-4 和图 5-5-5 所示。

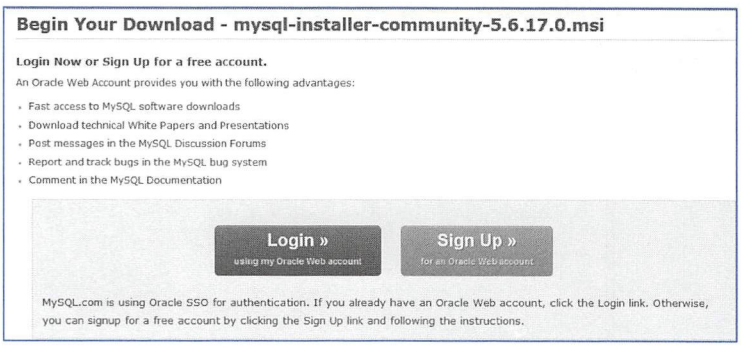


图 5-5-4　注册 Oracle 账号

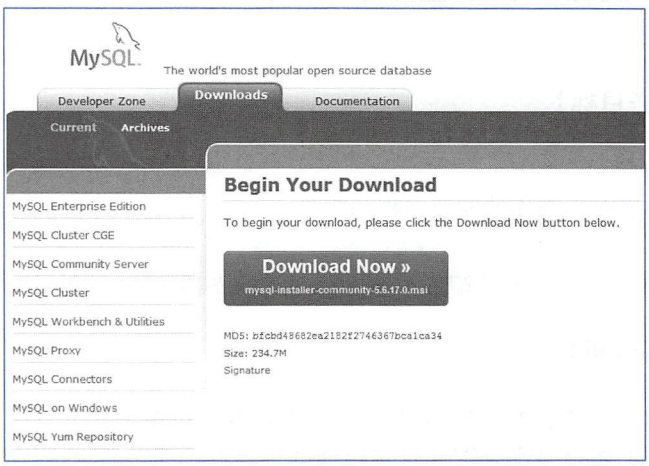


图 5-5-5　登录成功后下载

2. 安装 MySQL 数据库

① 双击打开 MySQL5.0 安装程序 MySQL\Setup.exe，单击“Next”按钮，如图 5-5-6 所示。

② 在选择安装类型的窗口中，有 Typical（默认）、Complete（完全）、Custom（用户自定义）3 个选项，选中“Custom”单选按钮，单击“Next”按钮，如图 5-5-7 所示。

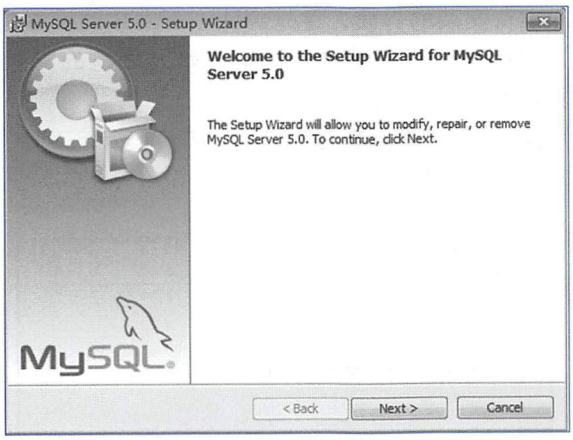


图 5-5-6　MySQL 安装向导

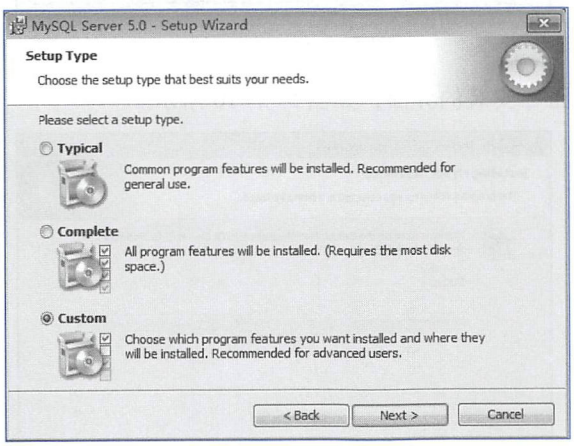


图 5-5-7　选择安装的类型

③ 在自定义安装界面中选择 MySQL 数据库的安装路径，本书设置的路径是 D:\Program File\MySQL，单击 Developer Components（开发者部分）前的倒三角按钮▼，在弹出的下拉列表中选择“This feature, and all subfeatures, will be installed on local hard drive.”选项，完成上述操作后单击“Next”按钮，如图 5-5-8 所示。

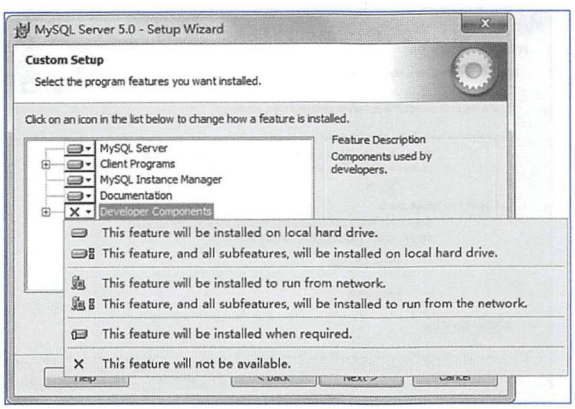


图 5-5-8　安装设置

④ 接下来进入准备安装的界面，首先确认一下刚才的设置是否正确，如果有误，单击“Back”按钮返回，可以重新设置。如果之前的配置没有错误，单击“Install”按钮继续安装，如图 5-5-9 所示。

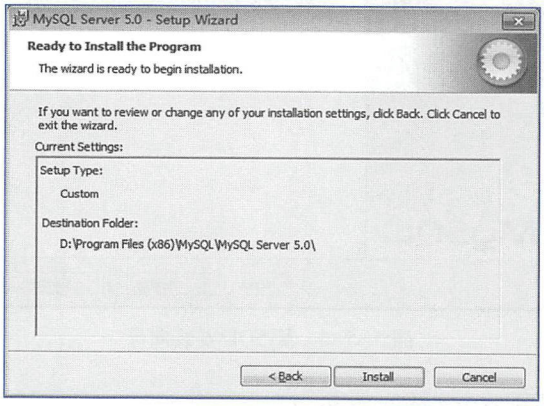


图 5-5-9　准备安装

⑤ 此时出现正在安装的界面，如图 5-5-10 所示。

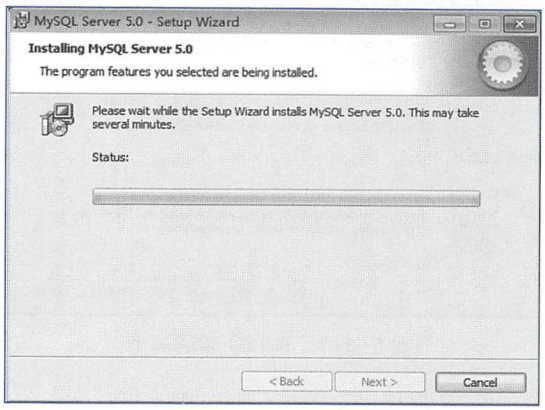


图 5-5-10　安装向导

⑥ 安装完成后出现注册界面，选中“Skip Sign-Up”单选按钮，单击“Next”按钮，如图 5-5-11 所示。

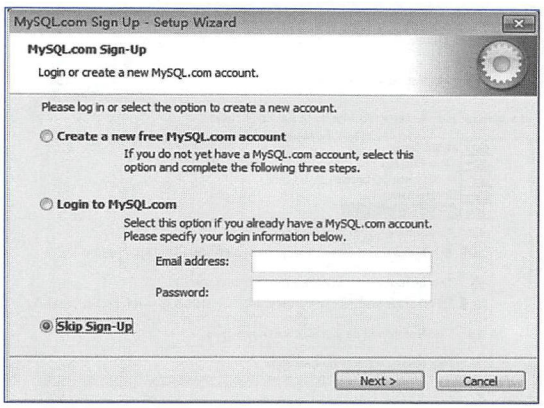


图 5-5-11　注册界面

⑦ 单击“Finish”按钮，完成 MySQL 安装，如图 5-5-12 所示。

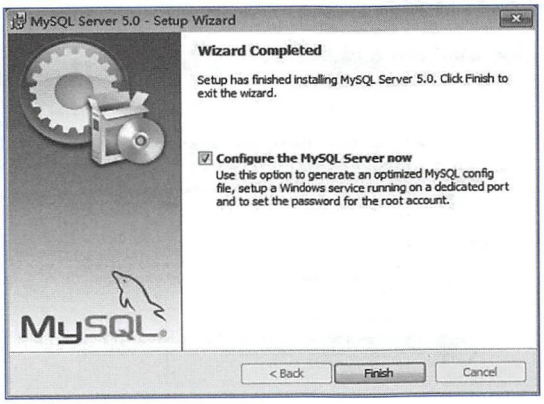


图 5-5-12　MySQL 安装完成

⑧ MySQL 数据库安装完成之后，出现如图 5-5-13 所示的配置界面向导，单击“Next”按钮。

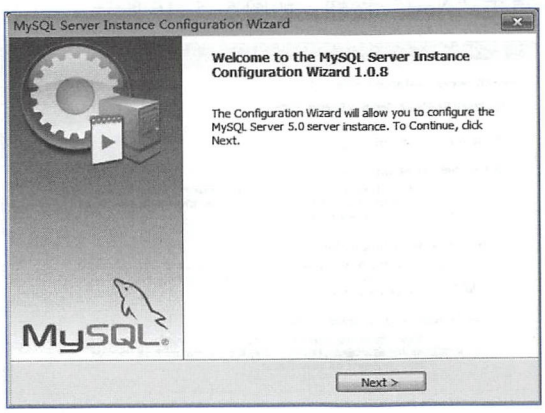


图 5-5-13　MySQL 配置参数向导

⑨ 配置方式选中“Detailed Configuration（详细配置）”单选按钮，单击“Next”按钮，如图 5-5-14 所示。

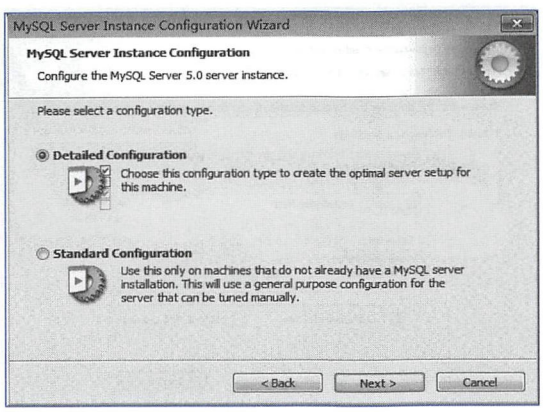


图 5-5-14　配置类型

⑩ 服务器类型选中“Developer Machine（开发测试类）”单选按钮，单击“Next”按钮，如图5-5-15所示。

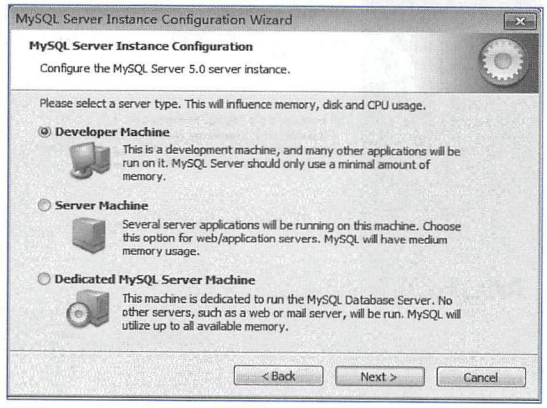


图5-5-15　服务器类型

⑪ 设置MySQL数据库的用途，选中“Multifunctional Database（通用多功能型）”单选按钮，单击“Next”按钮，如图5-5-16所示。

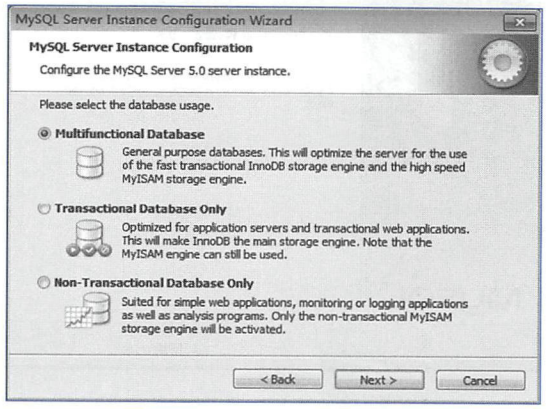


图5-5-16　数据库用途

⑫ 对InnoDB Tablespace进行配置，选择InnoDB数据库文件存储位置。选择默认安装目录Installation Path，单击“Next”按钮，如图5-5-17所示。

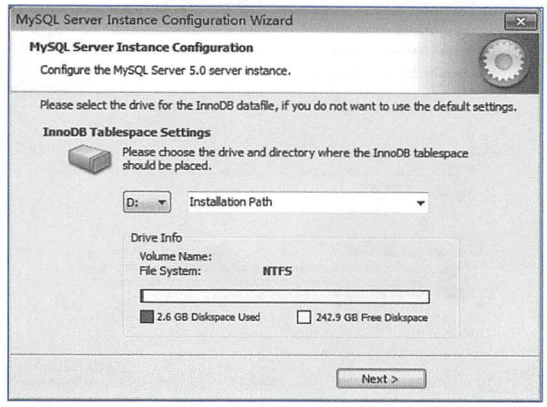


图5-5-17　存储位置

⑬ 配置 MySQL 服务器的参数，选中“Manual Setting（手动设置）”单选按钮，单击“Next”按钮，如图 5-5-18 所示。

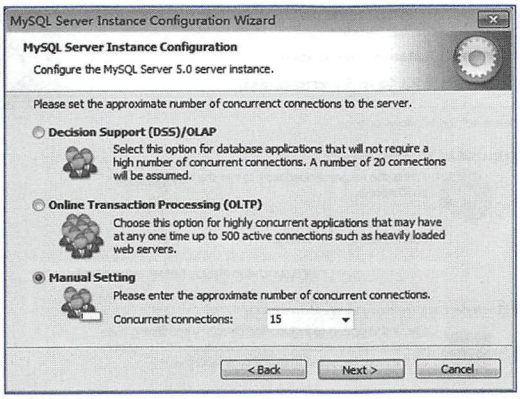


图 5-5-18　MySQL 并发参数的设置

⑭ 设置 TCP/IP 参数，选中“Enable TCP/IP Networking”单选按钮，默认的端口为 3306，选中“Enable Strict Mode”复选按钮，单击“Next”按钮，如图 5-5-19 所示。

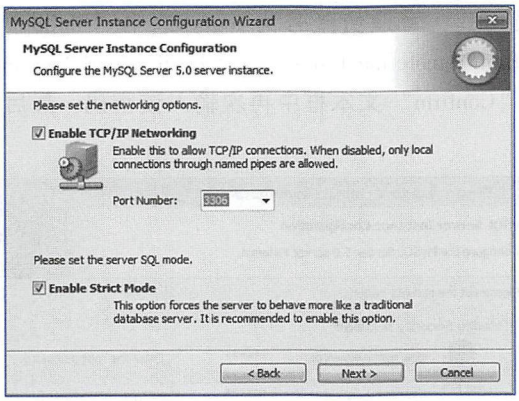


图 5-5-19　MySQL TCP 参数设置

⑮ 配置字符集配置，选中“Manual Selected Default Character Set/Collation”单选按钮，在“Character Set”下拉列表中选择“utf8”选项，单击“Next”按钮，如图 5-5-20 所示。

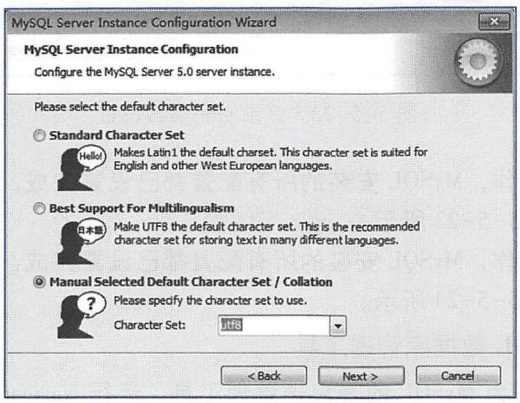


图 5-5-20　MySQL 的编码设置

⑯ 将 MySQL 安装为 Windows 服务，选中“Install As Windows Service”和“Include Bin Direction In Windows PATH”复选按钮，单击“Next”按钮，如图 5-5-21 所示。

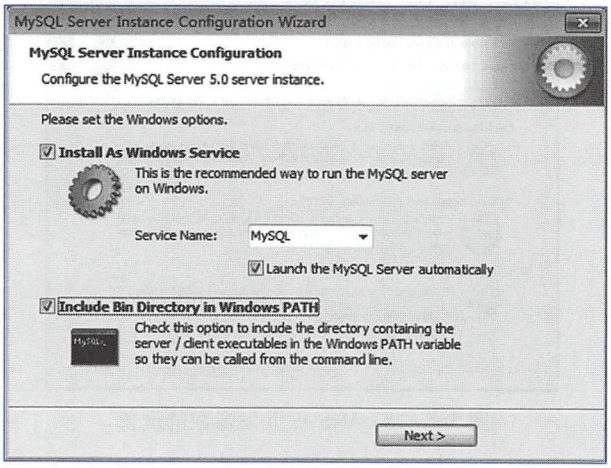


图 5-5-21　安装 Windows 服务

⑰ 设置管理员密码，管理员账号为 root。选中“Modify Security Settings”和“Enable root access from remote machines”复选按钮，在“New root password”文本框中输入密码，并在“Confirm”文本框中再次输入该密码，最后单击“Next”按钮，如图 5-5-22 所示。

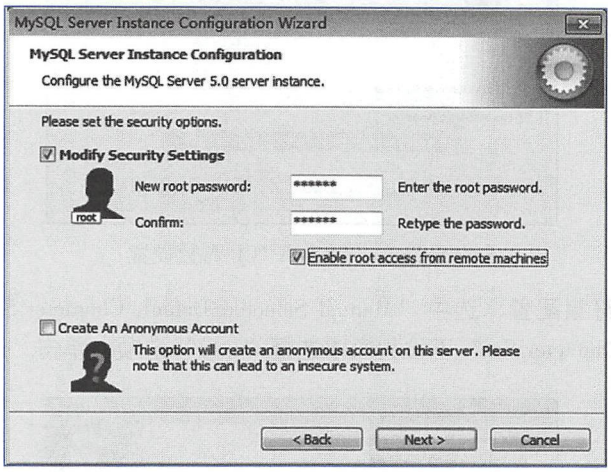


图 5-5-22　登录密码参数设置

⑱ 经过以上操作，MySQL 安装的所有配置都已设置完成，单击“Execute”按钮执行配置，如图 5-5-23 所示。

⑲ 经过以上操作，MySQL 安装的所有配置都已设置完成，单击“Execute”按钮执行配置，如图 5-5-24 所示。

3. 安装 MySQL 数据库管理工具

使用 Navicat 作为 MySQL 的客户端管理工具。运行 Navicat 8.0.exe 安装程序，设置安装目录后，单击“安装”按钮，如图 5-5-25 所示。

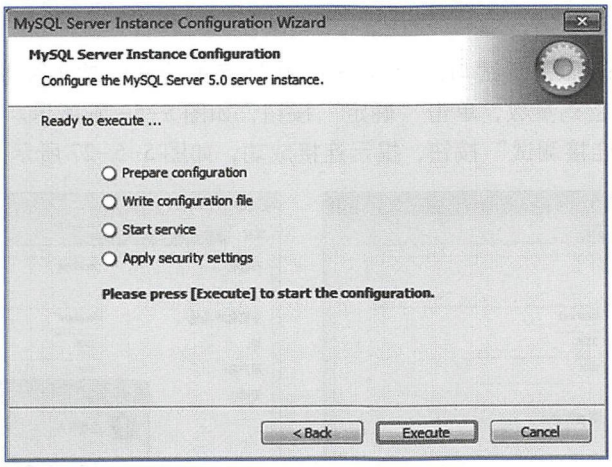


图 5-5-23　执行配置

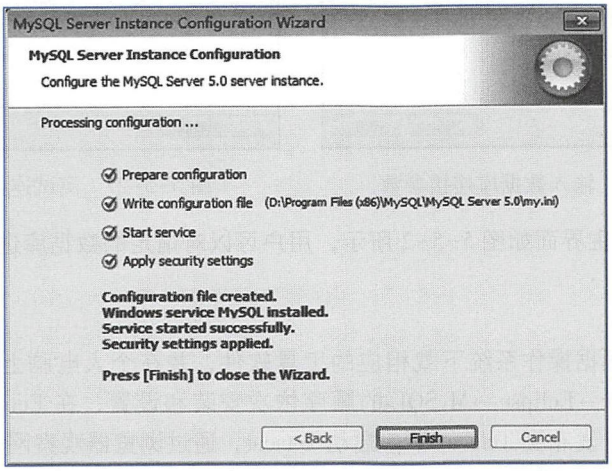


图 5-5-24　安装成功

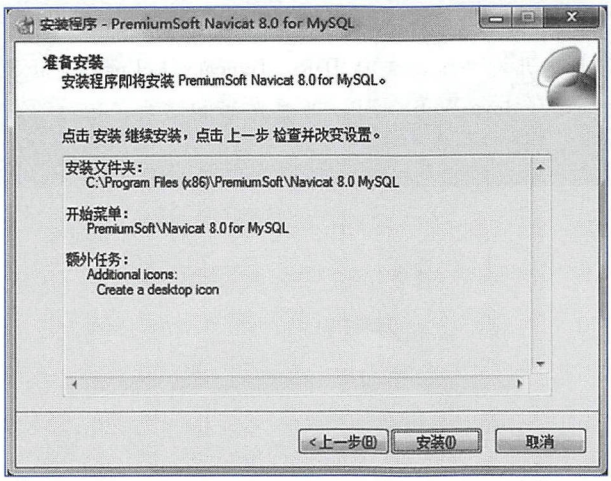


图 5-5-25　安装 MySQL 可视化工具

4. 功能验证测试

① 打开 Navicat，测试与本地 MySQL 数据库的连接，输入主机名、埠（端口号）、用户名与密码参数，单击“确定”按钮，如图 5-5-26 所示。

② 单击“连接测试”按钮，提示连接成功，如图 5-5-27 所示。

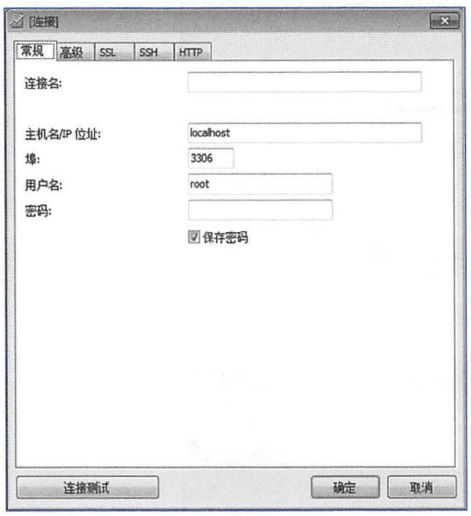


图 5-5-26　输入数据库连接参数

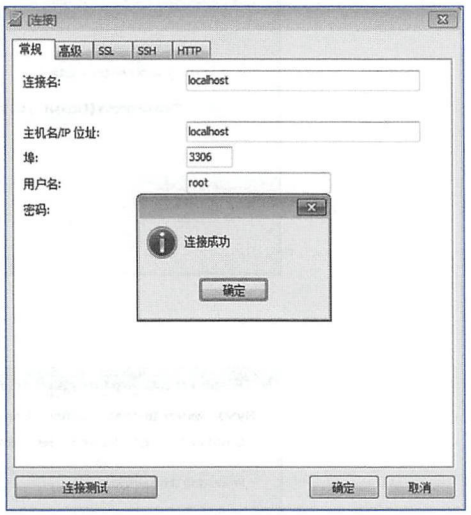


图 5-5-27　测试连接是否成功

③ Navicat 主界面如图 5-5-2 所示，用户可以对指定的数据库进行操作和管理。

技能训练

请在学生根据操作系统下载相应的工具软件，并在个人电脑上搭建开发环境：按 JDK→Tomcat→Eclipse→MySQL 的顺序依次安装和设置，在 Eclipse 中创建一个 Java Web 项目，发布到 Tomcat 中，启动 Tomcat，通过浏览器观察网页效果。

项目总结

在本项目中搭建了 Java Web 的开发环境，包括 JDK、Tomcat、Eclipse 和 MySQL 软件，并进行了相应的部署和配置，同时创建了 Java Web 项目，并且发布和运行，学生可以通过浏览器输入正确网址浏览网页。

项目 6　会员信息管理模块

PPT 会员信息管理模块

项目描述

SpringMVC+MyBatis 框架开发因其易上手和易掌握，且开发效率较高，非常受开发人员欢迎。 本项目通过会员信息管理模块的开发，详细讲解 SpringMVC +MyBatis 框架的搭建和功能开发，具体功能包括会员注册、登录、信息显示、修改等功能。

知识目标

- 了解 Bootstrap3 框架结构和开发方法。
- 了解 SpringMVC 框架的运行原理。
- 了解 MyBatis 框架的运行原理。
- 熟悉 SpringMVC+MyBatis 框架的搭建配置方法。
- 熟悉基于 SpringMVC+MyBatis 框架的开发方法。

技能目标

- 能通过 Bootstrap3 框架实现简单的网页开发。
- 能搭建配置 SpringMVC+MyBatis 开发框架。
- 能实现基于 SpringMVC+MyBatis 框架的开发。

任务列表

任务编号	任务名称	建议课时
任务 6.1	使用 Bootstrap 实现登录网页的快速开发	1
任务 6.2	采用 Ajax 技术实现简单的用户登录	3
任务 6.3	搭建 Spring+SpringMVC 框架实现用户简单注册	4
任务 6.4	整合 Spring+SpringMVC+MyBatis 框架实现会员信息更新	4
任务 6.5	上传头像图片	2
技能训练	基于 SSM 框架实现会员登录 基于 SSM 框架实现会员注册	2 2
	总计：	18 课时

任务 6.1　使用 Bootstrap 实现登录网页的快速开发

【任务描述】

通过 Bootstrap 前端网页开发框架实现登录页面，效果如图 6-1-1 所示。

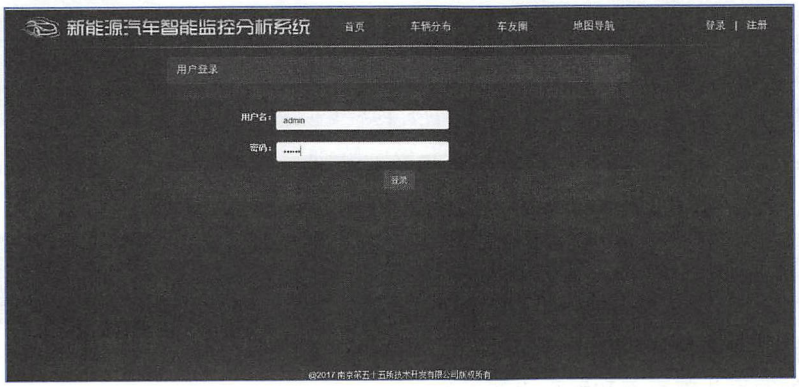


图 6-1-1　用户登录效果图

【任务目标】

微课 6-1 使用 Bootstrap 实现登录网页的快速开发

知识目标

- 了解 Bootstrap3 框架基本结构和使用方法。

技能目标

- 能够利用 Bootstrap3 实现网页表单效果。

【任务分析】

采用 Bootstrap 可以快速实现网页前台，利用 Bootstrap 中的表单相关样式实现登录、注册的页面效果。

【任务实施】

（1）新建登录页面 Login. jsp

在/WebContent/WEB-INF/page/ace/car 文件夹下新建文件 Login. jsp。登录界面采用 Bootstrap 技术来实现。

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
        <title>Insert title here</title>
        </head>
        <body>
        </body>
</html>
```

(2) 引入 CSS 和基础 JavaScript 文件

在<head></head>标签区域，引入相关 CSS 样式文件。

```
<link rel="stylesheet" href="res/bootstrap/css/bootstrap.min.css" />
<link rel="stylesheet" href="resources/css/style.css" />
<link rel="stylesheet" href="resources/css/page.css" />
```

在<body></body>标签区域，引入相关 js 文件，需要说明的是引用 Bootstrap 的 js 文件，必须同时引用 jQuery 的 js 文件。

```
<script src="resources/js/jquery-1.9.1.min.js" type="text/javascript"></script>
<script src="res/bootstrap/js/bootstrap.min.js" type="text/javascript"></script>
```

(3) 添加基本布局

添加网页头部页面和网页底部页面。

```
<body>
<script src="resources/js/jquery-1.9.1.min.js" type="text/javascript"></script>
<script src="res/bootstrap/js/bootstrap.min.js" type="text/javascript"></script>
<jsp:include page="../car/Head.jsp"></jsp:include>→网页头部
<!--主体部分-->
<%@ include file="Bottom.jsp"%>→网页底部
</body>
```

(4) 编写用户登录表单主体

① 列表组件用于以列表形式呈现复杂的和自定义的内容。创建一个基本的列表组，只需要向 元素添加 class . list-group 和 class . list-group-item 即可，代码如下。

```
<div class="list-group">
<div class="list-group-item">

</div>
</div>
```

② 在本实例中可以看到，除了列表主体部分以外还有一个列表标题，可以通过以下方式来添加列表标题：

- 使用“. list-group-item active class”向列表添加标题。
- 使用带有“. list-group-item-heading class”的<h1>-<h6>添加预定义样式的标题。

```
<div class="list-group">
<div class="list-group-item">
```

```
<a class="list-group-item active"><h4 class="list-group-item-heading">用户登录</h4></a>
<div class="list-group-item">
登录内容区域
</div>
</div>
</div>
```

这样就实现了一个简单的登录列表，如图 6-1-2 所示。

图 6-1-2　登录列表

接下来开始在该列表的内容区域加入登录内容，以支持用户输入用户名和密码等信息。

③ 登录内容实现步骤如下。

- 向 添加 ，新建一个默认 Bootstrap 网格。
- 把标签和控件放在一个带有 class . form-group 的 中。这是获取最佳间距所必需的。
- 向所有的文本元素<input>、<textarea>和<select>添加 class="form-control"。

```
<div class="list-group">
<a class="list-group-item active"><h4 class="list-group-item-heading">用户登录</h4></a>
<div class="list-group-item">
  <div class="row">
    <div class="col-md-12" style="background-color:#124E76;float:right;margin-top:20px;">
    <div class="form-group" style="margin-top:20px;margin-left: 12%;">
      <label class="form-label" style="color:#ffffff;">用户名:</label>
      <input type="text" id="username" class="form-control"
style="width:300px;margin-left:50px;">
    </div>
    <div class="form-group" style="margin-top:20px;margin-left: 12%;">
      <label class="form-label" style="color:#ffffff;">密码:</label>
      <input type="password" id="password" class="form-control"
style="width:300px;margin-left:50px;">
    </div>
  </div> <!-- end of col-md-12 div -->
  <div class="row" style="margin-top:10px;margin-bottom:10px;text-align:center;">
    <button type="button" style="text-align:center;" class="btn btn-primary">登录</button>
  </div>
</div> <!-- end of row div -->
</div> <!-- end of list-group-item div -->
</div> <!-- end of list-group div -->
```

④ 最后将登录内容嵌入到 list-group-item 区域中，用户登录界面如图 6-1-3 所示。

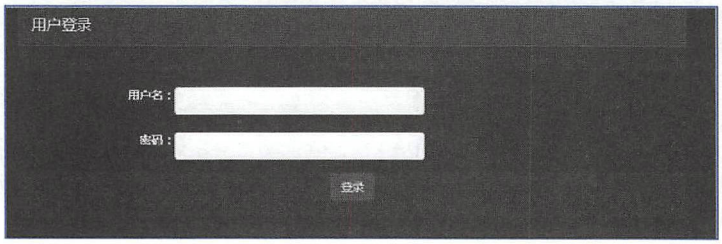
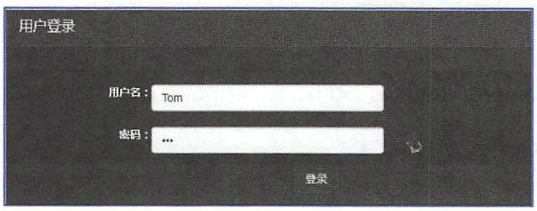


图 6-1-3　用户登录界面

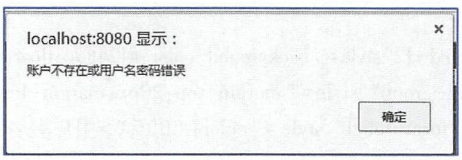
任务 6.2　采用 Ajax 技术实现简单的用户登录

【任务描述】

在任务 6.1 中，已经完成了登录界面的编写，接下来就要与后台进行数据交互，本任务采用 Ajax+JSON+Servlet 来实现数据交互，效果如图 6-2-1 所示。



(a)



(b)

图 6-2-1　用户登录 Ajax 交互效果

微课 6-2
采用 Ajax 技术实现简单的用户登录

【任务目标】

知识目标

- 掌握 Ajax 基本调用规则和方法。
- 了解 JSON 数据格式和数据传输。

技能目标

- 能够采用 Ajax、Servlet 技术实现异步数据交互。

【任务分析】

在登录界面 Login.jsp 输入用户名密码，如果用户名为 Tom、密码为 123，则登录成功，并显示“登录成功!”；否则登录失败，显示“账户不存在或用户名密码错误”。本任务重点讲解 Ajax 的数据交互实现，因此登录没有访问数据库。

【任务实施】

① 为“登录”按钮添加单击事件。

在 Login.jsp 页面中找到“登录”按钮相关代码，在代码中添加单击事件代码，代码如下。

```
<div class="row" style="margin-top: 10px; margin-bottom: 10px; text-align: center;">
    <button type="button" onclick="login()" style="text-align: center;" class="btn btn-primary">登录</button>
</div>
```

② 定义函数 login()。

在<body></body>中定义 login 函数。

```
1.  <script type="text/javascript">
2.      function login(){
3.          var username = $("#username").val();
4.          var password = $("#password").val();
5.          var data = {
6.              username : username,
7.              password : password
8.          }
9.          $.ajax({
10.             type : "post",
11.             url : '<%=path%>' + "/LoginServlet",
12.             data : data,
13.             dataType : "jsonp",//数据类型为 json
14.             jsonp:"jsoncallback",
15.             success : function(data){
16.                 if (data == "true") {
17.                     alert("登录成功!");
18.                 } else {
19.                     alert("账户不存在或用户名密码错误!");
20.                 }
21.             }
22.         })
23.     }
24. </script>
```

代码说明：

- 第 3 行～第 4 行：定义变量分别存储用户名和密码两个文本框的值，$("#username") 中#表示通过 id 查找对象。
- 第 5 行～第 8 行：将 username 和 password 两个变量定义在一个 data 对象中，这种方式便于传递多个参数。

● 第9行~第22行：$ajax()方法调用需要设置很多参数，其中type表示请求方式为post，url为请求地址，data传输数据参数的值就是在第5行~第8行所定义的变量data，dataType表示数据类型为jsonp格式，success表示请求成功后调用的回调函数，如果成功则提示“登录成功!”，否则显示“账户不存在或密码错误!”。

③ 定义登录后台控制类LoginServlet.java。

在src源码文件夹中找到包com.piesat.zyms.web.cms，新建Servlet——LoginServlet.java。核心方法doPost()代码如下。

```
1.  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
2.  ServletException, IOException {
3.      String username = request.getParameter("username");
4.      String password = request.getParameter("password");
5.      String jsonp = request.getParameter("jsoncallback");
6.      String result = "";
7.      if(username.equals("tom")&&password.equals("abc123")){
8.          result=new Gson().toJson("true");
9.      }else {
10.         result=new Gson().toJson("false");
11.     }
12.     response.setCharacterEncoding("UTF-8");
13.     response.setContentType("text/html");
14.     if(jsonp != null){
15.         result = jsonp+"("+result+")";
16.     }
17.     response.getWriter().write(result);
18. }
```

代码说明：

● 第3行~第5行：接收参数username、password、jsoncallback。

● 第7行~第11行：如果用户名和密码验证通过，将true转换为json格式存储在变量result中，否则验证失败，将false转换为json格式存储在变量result中。

● 第14行~第16行：如果jsoncallback不为null，则在结果result变量前增加jsoncallback的值。

● 第17行：将结果响应给前端。

总结分析：当单击“登录”按钮时，触发login函数，向LoginServlet发送请求，并传递相关参数数据，在LoginServlet逻辑处理完成后将结果返回到login函数中，并显示相应结果。特别注意：Login函数代码中第15行的data参数，就是LoginServlet中返回的result值。

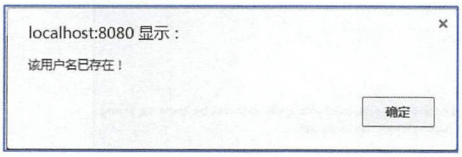
至此为止，通过简单的Ajax+Servlet实现的用户登录功能已经完成，重启Tomcat，打开浏览器，输入网址http://localhost:8080/car/energy/login，然后按Enter键，进行访问并测试。

任务 6.3　搭建 Spring+SpringMVC 框架实现会员简单注册

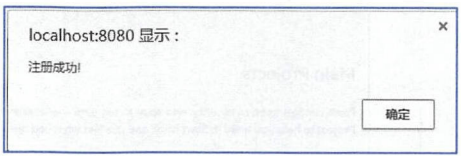
【任务描述】

在本任务中搭建 Spring+SpringMVC 框架，并结合 Ajax 技术实现简单的用户注册，输入用户信息验证通过后提交后台处理程序。如果用户名为 Tom，则提示“该账号已注册”，否则提示“注册成功”，如图 6-3-1 所示。通过简单的用户注册重点讲解 Spring+SpringMVC 框架的搭建和开发过程，暂时不涉及数据库。

(a)



(b)



(c)

图 6-3-1　用户注册界面

【任务目标】

知识目标

- 了解 Spring 框架的基本结构。
- 了解 SpringMVC 框架运行原理和处理流程。
- 掌握 SpringMVC 配置文件的基本格式规范。
- 掌握 SpringMVC 注解规范。

技能目标

- 能够搭建 Spring+SpringMVC 框架体系。
- 能够 Spring+SpringMVC 框架下进行简单的功能开发。
- 能够结合 Ajax 技术和 Spring+SpringMVC 框架技术进行简单的开发。

微课 6-3
搭建 Spring+SpringMVC 框架实现会员简单注册

【任务分析】

简单的用户注册功能流程是用户输入基本信息后，单击“注册”按钮，将表单数据通过 Ajax 提交给 SpringMVC 框架中的 Controller 进行处理。Controller 将结果返

回给注册页面并进行相应的提示。在这个开发过程中一共分为以下 3 个重要环节：

- Spring+SpringMVC 框架搭建环节。
- Spring+SpringMVC 架构下的代码开发环节。
- 编写注册页面中$. ajax( ) 方法的环节。

【任务实施】

(1) 下载 Spring 相关 jar 包

① 进入 Spring 官网并单击导航栏中的“PROJECTS”按钮，如图 6-3-2 所示。

图 6-3-2　Spring 官网

② 选择“SPRING FRAMEWORK”项目，单击图标进入，如图 6-3-3 所示。

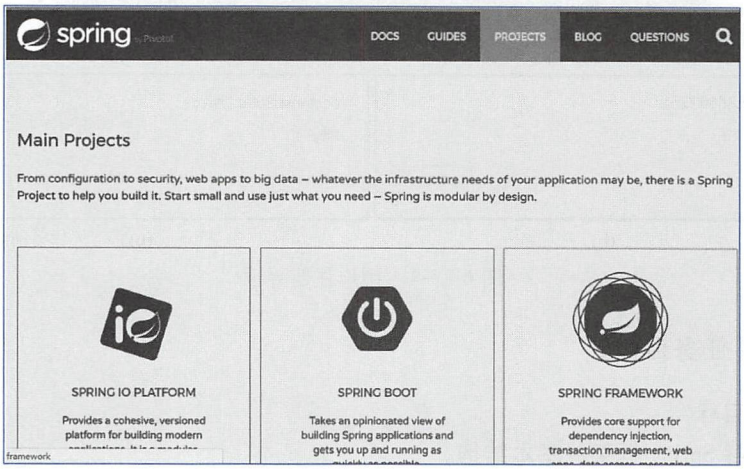


图 6-3-3　项目页面

③ 找到要下载的版本（本任务采用 4. 3. 12 版本），单击对应的“Reference”按钮，如图 6-3-4 所示。

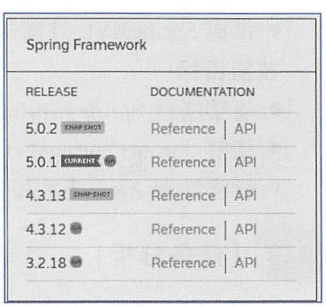


图 6-3-4　Spring 版本列表

④ 在 Distribution Zip Files 下，找到下载地址 http://repo. spring. io/release/org/springframework/spring，单击文字超链接进入，如图 6-3-5 所示。

⑤ 找到对应的版本进行下载，如图 6-3-6 所示。

⑥ 将下载到的 jar 包文件复制到 WEB-INF/lib 文件夹中，如图 6-3-7 所示。

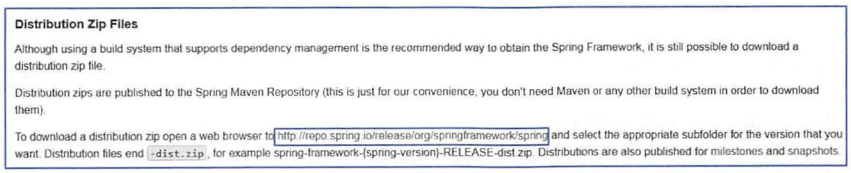


图 6-3-5　下载链接

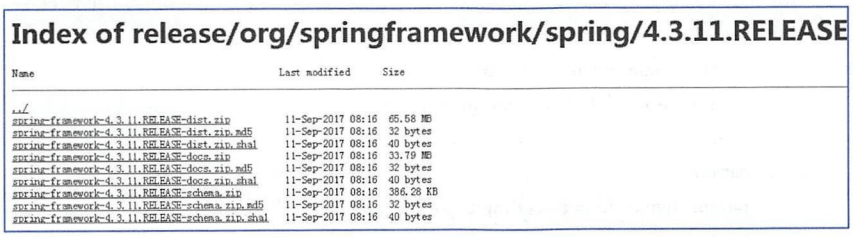


图 6-3-6　下载页面

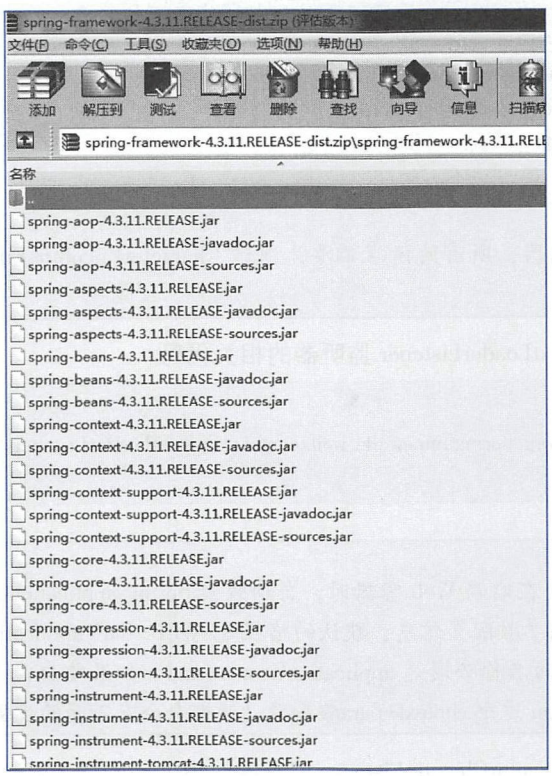


图 6-3-7　SpringMVC 中 jar 包列表

(2) 在 web.xml 文件中配置 Spring

打开 WEB-INF 文件夹下的 web.xml 文件进行编辑，以下代码添加在<web-app></web-app>之间。

① 添加参数 display-name 配置代码。

```
<display-name>car</display-name>
```

注意

display-name 定义了 Web 应用的名称。

② 添加中文编码相关配置代码。

```
<filter>
    <filter-name>SpringEncodingFilter</filter-name>
    <filter-class>org. springframework. web. filter. CharacterEncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<filter-mapping>
    <filter-name>SpringEncodingFilter</filter-name>
    <url-pattern>/ * </url-pattern>
</filter-mapping>
```

注意

本段代码配置的是编码过滤器，所有的请求都要先通过 CharacterEncodingFilter 过滤器，编码方式是“UTF-8”。

③ 添加 ContextLoaderListener 监听器的相关代码。

```
<listener>
<listener-class>org. springframework. web. context. ContextLoaderListener</listener-class>
</listener>
```

注意

ContextLoaderListener 的作用是在启动 Web 容器时，自动装配 Spring applicationContext. xml 的配置信息。如果在 web. xml 中不写任何参数配置信息，默认的路径是/WEB-INF/applicationContext. xml，在 WEB-INF 目录下创建的 xml 文件的名称必须是 applicationContext. xml。如果要自定义文件名，可以在 web. xml 里加入 contextConfigLocation 这个 context-param 参数（此部分会在下面的步骤⑤中详细讲解）。

④ 添加 IntrospectorCleanupListener 监听器的相关代码。

```
<listener>
<listener-class>org. springframework. web. util. IntrospectorCleanupListener</listener-class>
</listener>
```

注意

Spring 中提供了一个名为 org. springframework. web. util. IntrospectorCleanupListener 的监听器，其主要负责处理由于 JavaBeans Introspector 的使用而引起的缓冲泄露。

⑤ 添加前置控制器 Servlet 配置代码。

```
<servlet>
<servlet-name>CmsAdmin</servlet-name>
<servlet-class>org. springframework. web. servlet. DispatcherServlet</servlet-class>
<init-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/config/cms-servlet-admin. xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>CmsAdmin</servlet-name>
<url-pattern>/</url-pattern>
</servlet-mapping>
```

注意

DispatcherServlet 拦截匹配的请求中，Servlet 拦截匹配规则要自己定义，把拦截下的请求依据某某规则分发到目标 Controller（这里写的 Action）来处理。在 DispatcherServlet 的初始化过程中，框架会在 Web 应用的 WEB-INF 文件夹下寻找名为[servlet-name]-servlet. xml 的配置文件，生成文件中定义的 bean，在本段代码中就找到 cms-servlet-admin. xml 文件。

load-on-startup：web. xml 中可配置多个 servlet，load-on-startup 可指定在系统启动时按顺序加载 servlet。

url-pattern：表示哪些请求交给 Spring Web MVC 处理。此处会处理所有 URL 为"/*"的请求。

(3) 配置 SpringMVC 文件 cms-servlet-admin. xml

① 在 WEB-INF/config 文件夹下新建一个 xml 文件，命名为 cms-servlet-admin. xml。

② 添加命名空间代码，接下来的代码都是在<beans></beans>中添加。

```
<beans xmlns="http://www. springframework. org/schema/beans"
    xmlns:xsi="http://www. w3. org/2001/XMLSchema-instance"
    xmlns:context="http://www. springframework. org/schema/context"
    xmlns:mvc="http://www. springframework. org/schema/mvc"
    xsi:schemaLocation="http://www. springframework. org/schema/beans
http://www. springframework. org/schema/beans/spring-beans-3. 2. xsd
    http://www. springframework. org/schema/context
http://www. springframework. org/schema/context/spring-context-3. 2. xsd
    http://www. springframework. org/schema/mvc
http://www. springframework. org/schema/mvc/spring-mvc-3. 2. xsd"
    default-lazy-init="true">
```

③ 添加代码启动 Spring MVC 的注解功能，完成请求和注解 POJO 的映射。解决@ResponseBody乱码问题，需要在 annotation-driven 之前，否则乱码问题同样无法解决。

```
<bean
class="org. springframework. web. servlet. mvc. method. annotation. RequestMappingHandlerAdapter">
<property name="messageConverters"></beans>
```

```
<list>
<bean class="org.springframework.http.converter.StringHttpMessageConverter">
<property name="supportedMediaTypes">
<list>
<value>text/html;charset=UTF-8</value>
<value>text/plain;charset=UTF-8</value>
</list>
</property>
</bean>
<bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter">
        <property name="supportedMediaTypes">
        <list>
            <value>application/json</value>
            </list>
        </property>
        </bean>
</list>
</property>
</bean>
```

④ 添加代码，使用组件扫描的方式可以一次扫描多个 Controller。

```
<context:component-scan base-package="com.piesat.zyms.web" />
```

⑤ 添加代码，把标记了@ Controller 注解的类转换为 bean。

```
<context:component-scan base-package="com.piesat.zyms.service" />
```

⑥ 添加代码，要使用 Spring MVC 中的@ Controller 注解，就必须要配置<mvc:annotation-driven />，否则 org.springframework.web.servlet.DispatcherServlet 无法找到控制器，并把请求分发到控制器。

```
<mvc:annotation-driven/>
<mvc:default-servlet-handler />
```

其余代码在此不做详细解释，可直接拷贝资源包 cms-servlet-admin.xml 配置文件来使用。

(4) 配置 applicationContext.xml 文件

① 在 WEB-INF/config 文件夹下新建一个 xml 文件，命名为 applicationContext.xml。

② 添加命名空间代码。

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd
    http://www.springframework.org/schema/mvc
```

```
http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd"
    default-lazy-init="true">
</beans>
```

因为目前还不涉及到数据库连接，本任务中 applicationContext.xml 中数据库配置相关代码暂时不做配置。

(5) 创建控制器 IndexController.java

① 在 src 文件中新建包 com.piesat.zyms.web.cms。

② 在步骤①创建的包中，新建一个 Class 命名为 IndexController.java，在类头部添加注解代码@Controller 和@ RequestMapping("/energy")，代码如下。

```
1.  @Controller
2.  @RequestMapping("/energy")
3.  public class IndexController {
……
4.  }
```

代码说明：

- 第 1 行：@Controller 用于标记在一个类上，使用它标记的类就是一个 SpringMVC Controller 对象。分发处理器将会扫描使用了该注解的类的方法，并检测该方法是否使用了@RequestMapping 注解。@Controller 只是定义了一个控制器类，而使用@RequestMapping 注解的方法才是真正处理请求的处理器。
- 第 2 行：@RequestMapping 是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

③ 在类内部，创建方法 adduser，用于用户注册，代码如下。

```
1.  @RequestMapping("/adduser")
2.  public void addUser(HttpServletRequest request,HttpServletResponse response, ModelMap
model) throws IOException{
3.      String username = request.getParameter("username");
4.      String result = "";
5.      String jsonp = "";
6.      if(username.equals("tom")){
7.          result=new Gson().toJson("false");
8.      }else{
9.          result=new Gson().toJson("true");
10.     }
11.     jsonp = request.getParameter("jsoncallback");
12.     response.setCharacterEncoding("UTF-8");
13.     response.setContentType("text/html");
14.     if(jsonp != null)
15.         result = jsonp+"("+result+")";
```

```
16.     response.getWriter().write(result);
17.   }
```

代码说明：

第6行~第10行：根据任务需求，如果用户名为tom，则返回结果为false，否则返回结果为true，结果返回到调用Ajax方法的Registe.jsp页面再进行处理。

④ 在类内部，创建方法registe，用于访问注册页面，代码如下。

```
@RequestMapping(value = "/registe", method = RequestMethod.GET)
public ModelAndView registe(HttpServletRequest request,HttpServletResponse response, ModelMap model) {
    ModelAndView mv = new ModelAndView();
    mv.setViewName("car/Registe.jsp");
    return mv;
}
```

(6) 编辑Registe.jsp中的Ajax调用方法

① 通过Bootstrap编辑页面在任务6.1中已经详细阐述过，本任务则直接使用已完成的注册页面Registe.jsp。

② 在Registe.jsp页面添加save()函数，用于异步调用IndexController中的addUser方法。

```
<div class="row" style="margin-top:10px;margin-bottom:10px;text-align:center;">
<button type="button" onclick="save()" style="text-align:center;" class="btn btn-primary">
注册
</button>
</div>
```

```
function save(){
    var username=$("#username").val();
    var password=$("#password").val();
    var nickname=$("#nickname").val();
    var y1=$("#year").val();
    var m1=$("#month").val();
    var d1=$("#day").val();
    var y2=$("#years").val();
    var m2=$("#months").val();
    var d2=$("#days").val();
    var val=$('input:radio[name="sex"]:checked').val();
    var item = $('input:checkbox[name="com"]:checked').val();
    if(item!=1)
        item=0;
```

```
        var seperator = "-";
        var birthdate = y1 + seperator + m1 + seperator + d1;
        var dltime = y2 + seperator + m2 + seperator + d2;
        $.ajax({
            type : "post",
            url : '<%=path %>' + "/energy/adduser",
            data : {
                username : username,
                password : password,
                nickname : nickname,
                val : val,
                item : item,
                birthdate : birthdate,
                dltime : dltime,
            },
            dataType : "jsonp",//数据类型为 json
            jsonp:"jsoncallback",
            success : function(data){
                if(data==="false"){
                    alert("该用户名已存在!");
                }else{
                    alert("注册成功!");
                window.location.href="http://" + location.host + "<%=path %>/" +"ener-
gy/login";
                }
        }
        });
    }
```

代码部分结构与用户登录中的 Ajax 方法相似，在此不再赘述。

至此为止，SpringMVC 模式下未连接数据库的用户注册功能已经完成，重启 Tomcat，打开浏览器，输入网址 http://localhost:8080/car/energy/registe，按 Enter 键，进行访问并测试。

任务 6.4　整合 Spring+SpringMVC+MyBatis 实现会员信息更新

【任务描述】

任务 6.3 实现了简单的会员信息注册，但并没有真正写入数据库，在本任务中通过整合 Spring+SpringMVC+MyBatis 框架实现会员信息的数据库更新。用户登录成功后，单击用户的下拉菜单，在下拉菜单中选择“基本信息”菜单命令，打开会员基本信息页面，编辑用户信息后单击“保存”按钮，实现会员信息的更新，如图 6-4-1 所示。

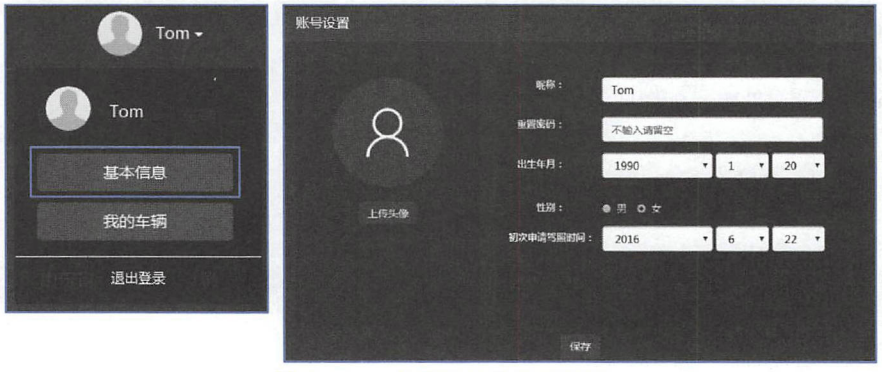


图 6-4-1　会员信息的显示更新

【任务目标】

知识目标

- 了解 MyBatis 框架架构和原理。
- 了解 Spring+SpringMVC+MyBatis 框架的开发方法。

技能目标

- 能搭建 Spring+SpringMVC+MyBatis 框架并进行功能开发。

微课 6-4
整合 Spring+SpringMVC+MyBatis 实现会员登录

【任务分析】

会员信息更新的流程是，先根据用户名从数据库中读出会员信息并显示，修改好后单击“保存”按钮将数据更新到后台数据库。

注意

SpringMVC 的相关配置已经在任务 6.3 中详细说明，本节不再赘述。

在任务 6.3 的基础上具体的开发流程如下：

① 下载 MyBatis 相关 jar 包。

② MyBatis 配置，包括 jdbc.properties、applicationContext.xml、SqlMapConfig.xml。

③ 创建会员信息数据对应 POJO 类 User.java，定义数据结构，其与表结构一致。

④ 用户显示的实现：

- 在 IndexController 中实现一个方法，用于将请求转发至 Accset.jsp。
- 在视图层 Accset.jsp 页面，编写 Ajax 方法实现用户信息数据的显示。
- 在 UserController 中实现 accset 方法，用于数据交互。

⑤ 用户信息更新的实现。

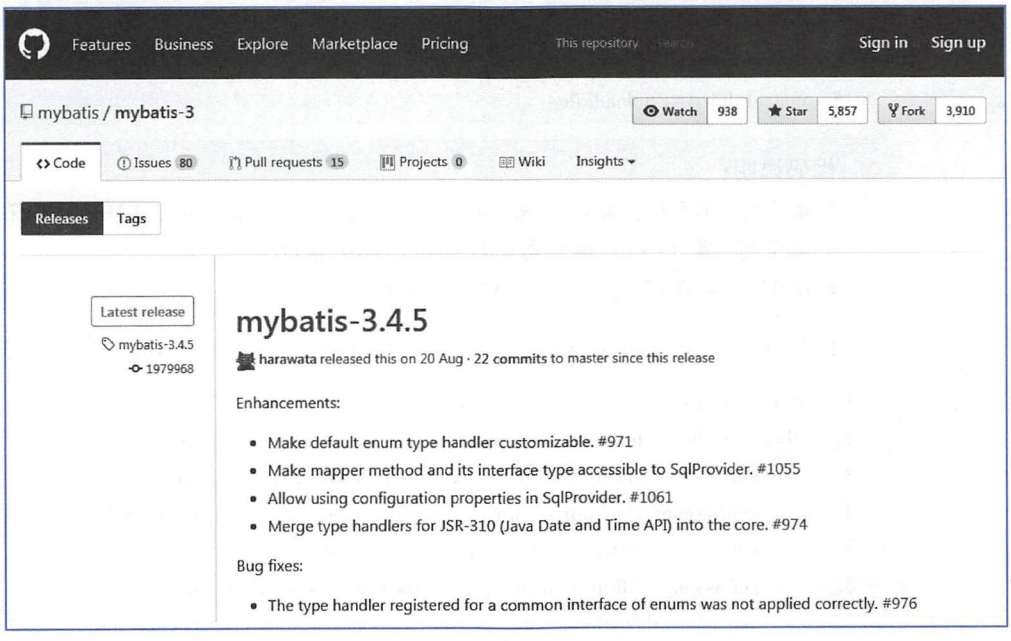
- 在 Accset.jsp 页面单击“保存”按钮触发信息保存方法 save()。
- 编辑控制层 UserController.java，添加 updateUser()方法。
- 创建业务逻辑的处理层类 UserService.Java，声明 updateUser 方法。
- 创建对象持久化映射层类 UserMapper.java、UserMapper.xml，声明和定义会员表上更新方法 updateUser()。

微课 6-5
会员注册和信息更新

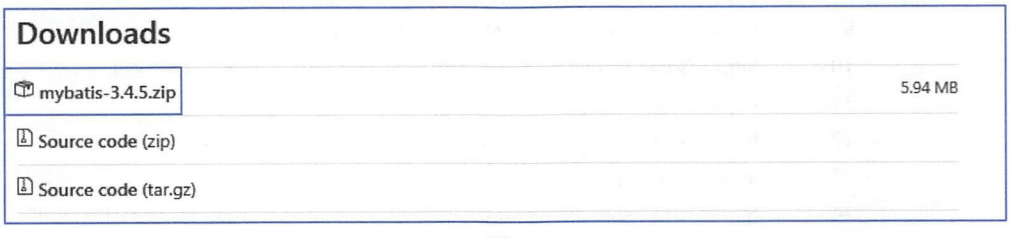
【任务实施】

(1) 下载 MyBatis 相关 jar 包

MyMatis 的核心包只有一个 mybatis-3.x.0.jar，另外还有一些可选的依赖包(日志、代理等所需要的资源)，在下载的压缩包中可以找到。下载地址 https://github.com/mybatis/mybatis-3/releases，如图 6-4-2 所示。



(a)



(b)

图 6-4-2　MyBatis 下载界面

将下载到的 jar 文件复制到 WEB-INF/lib 文件夹下。

(2) MyBatis 配置

① 在 WEB-INF\config 文件夹中新建文件 jdbc.properties，代码如下。

```
1. #config database
2. database.driverClassName=com.mysql.jdbc.Driver
3. database.url=jdbc:mysql://localhost:3306/energycar? autoReconnect=true&useUnicode
   =true&characterEncoding=UTF-8&zeroDateTimeBehavior=convertToNull
4. database.username=root
5. database.password=888888
```

```
6.  #config collection pool
7.  pool.initialPoolSize=1
8.  pool.minPoolSize=0
9.  pool.maxPoolSize=2
10. pool.maxIdleTime=120
11. pool.acquireIncrement=1
12. uploadPath=/u/cms/www
13. portalDir=/WEB-INF/cms_sys/portal
14. imageFilePath=D\:/file/
15. imageFileUrl=/uploadFile/
```

代码说明：

- 第2行~第5行：配置数据库连接相关属性，包括驱动、url、用户名和密码。
- 第7行~第11行：配置数据库连接池相关属性。
- 第12行~第15行：文件上传路径配置。

② 在 WEB-INF\config 文件夹中新建文件 applicationContext.xml，代码如下。

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <beans xmlns="http://www.springframework.org/schema/beans"
3.      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.      xmlns:context="http://www.springframework.org/schema/context"
5.      xmlns:tx="http://www.springframework.org/schema/tx"
6.      xmlns:aop="http://www.springframework.org/schema/aop"
7.      xsi:schemaLocation="http://www.springframework.org/schema/aop
8.      http://www.springframework.org/schema/aop/spring-aop-4.1.xsd
9.      http://www.springframework.org/schema/beans
10.     http://www.springframework.org/schema/beans/spring-beans.xsd
11.     http://www.springframework.org/schema/tx
12.     http://www.springframework.org/schema/tx/spring-tx-4.1.xsd
13.     http://www.springframework.org/schema/context
14.     http://www.springframework.org/schema/context/spring-context-4.1.xsd"
15.     default-lazy-init="true">
    <!--使用注解注入 properties 中的值-->
16.  <beanid="config" class="org.springframework.beans.factory.config.Properties
    Factory Bean">
17.     <property name="locations">
18.         <value>/WEB-INF/config/jdbc.properties</value>
19.     </property>
20. </bean>
    <!--使用注解直接注入 properties 文件中的配置-->
21. <bean id="propertyConfigurer" class="org.springframework.beans.factory.config.
    PreferencesPlaceholderConfigurer">
22.     <property name="properties" ref="config"></property>
```

```
23.  </bean>
     <! --配置数据源 c3p0 数据库连接池 -->
24.  <bean id="dataSource"    class="com.mchange.v2.c3p0.ComboPooledDataSource">
25.  <property name="driverClass" value="${database.driverClassName}"/>
26.  <property name="jdbcUrl" value="${database.url}"/>
27.  <property name="user" value="${database.username}"/>
28.  <property name="password" value="${database.password}"/>
29.  <property name="initialPoolSize" value="${pool.initialPoolSize}"/>
30.  <property name="minPoolSize" value="${pool.minPoolSize}"/>
31.  <property name="maxPoolSize" value="${pool.maxPoolSize}"/>
32.  <property name="maxIdleTime" value="${pool.maxIdleTime}"/>
33.  <property name="acquireIncrement" value="${pool.acquireIncrement}"/>
34.  </bean>
     <! -- jdbc 事务管理器 -->
35.  <bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSource-
     TransactionManager">
36.  <property name="dataSource" ref="dataSource" />
37.  </bean>
     <! --配置 sqlSessionFactory -->
38.  <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
39.  <! --数据库连接池 -->
40.      <property name="dataSource" ref="dataSource"></property>
41.      <property name="typeAliasesPackage" value="com.piesat.zyms.domain">
     </property>
42.  <! --加载 Mybatis 全局配置文件 -->
43.  <property name="configLocation"
         value="/WEB-INF/config/SqlMapConfig.xml"></property>
44.  </bean>
     <! --配置 mapper 扫描器 -->
45.  <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
46.      <property name="basePackage" value="com.piesat.zyms.persistence"></property>
47.  </bean>
48.   <bean id="lobHandler"  class="org.springframework.jdbc.support.lob.
      DefaultLobHandler" lazy-init="true"/>
</beans>
```

代码说明：

- 第 2 行~第 15 行：声明命名空间。
- 第 16 行~第 20 行：将 jdbc.properties 数据库配置文件注入 PropertiesFactoryBean 中，生成 id 为 config 的配置对象。
- 第 21 行~第 23 行：使用注解直接注入 properties 文件中的配置，注意 config 对象的引用。

- 第 24 行~第 34 行：根据配置文件 properties 文件中的内容配置数据源 c3p0 数据库连接池。
- 第 35 行~第 37 行：开启 JDBC 事务处理。
- 第 38 行~第 44 行：配置 sqlSessionFactory（MyBatis 调用的入口），包括数据源、关联类的包、配置文件等。
- 第 45 行~第 47 行：配置 mapper 扫描器，扫描 com. piesat. zyms. persistence 包。
- 第 48 行：当要处理的对象的数据库字段类型是 blob 时，指定 lobHandler，对于 MySQL、DB2、MS SQL Server、Oracle 10g，使用 DefaultLobHandler 即可。

③ 在 WEB-INF\config 文件夹中新建文件 SqlMapConfig. xml，代码如下。

```
1.  <?xml version="1.0" encoding="UTF-8" ?>
2.  <!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
      "http://mybatis.org/dtd/mybatis-3-config.dtd">
3.  <configuration>
4.  <! --配置分页插件 -->
5.  <plugins>
6.  <plugin interceptor="com.github.pagehelper.PageHelper">
7.  <! --设置数据库类型 Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL 六种数据库-->
8.  <property name="dialect" value="mysql"/>
9.  </plugin>
10. </plugins>
11. </configuration>
```

代码说明：

- 第 6 行~第 8 行：配置 MyBatis 物理分页插件 PageHelper，连接 MySQL 数据库。

④ 在 WEB-INF\web. xml 文件中添加参数 context-param 配置代码。

```
<context-param>
<param-name>contextConfigLocation</param-name>
<param-value>/WEB-INF/config/applicationContext.xml</param-value>
</context-param>
```

注意

context-param 元素声明应用范围内的初始化参数，param-name 声明参数名，param-value 声明参数的值。这段代码声明了初始化参数都在配置文件/WEB-INF/config/applicationContext. xml 中。

(3) 创建会员信息表对应的 POJO 类 User. java

User 类是对 user 表的一个映射，类的属性与表的字段一一对应，如图 6-4-3 所示。

① 在 src 包中，新建一个包 com. piesat. zyms. domain. core。

② 在步骤①创建的包中，新建一个类 User，类的属性如图 6-4-3（b）所示。

③ 创建好属性后，另起一行，右击，在弹出的快捷菜单中选择“source→Generate Getters and Setters”命令，如图 6-4-4 所示。

名	类型	长度
id	varchar	255
username	varchar	255
password	varchar	255
createtime	datetime	0
nickname	varchar	255
birthdate	date	0
sex	int	4
dltime	date	0
hpic	varchar	255
usertype	int	255

(a)

```
public class User {

    private String id;
    private String username;
    private String password;
    private Date createtime;
    private String nickname;
    private Date birthdate;
    private int sex;
    private Date dltime;
    private String hpic;
    private int usertype;
```

(b)

图 6-4-3　user 表字段与 User 类属性对应图

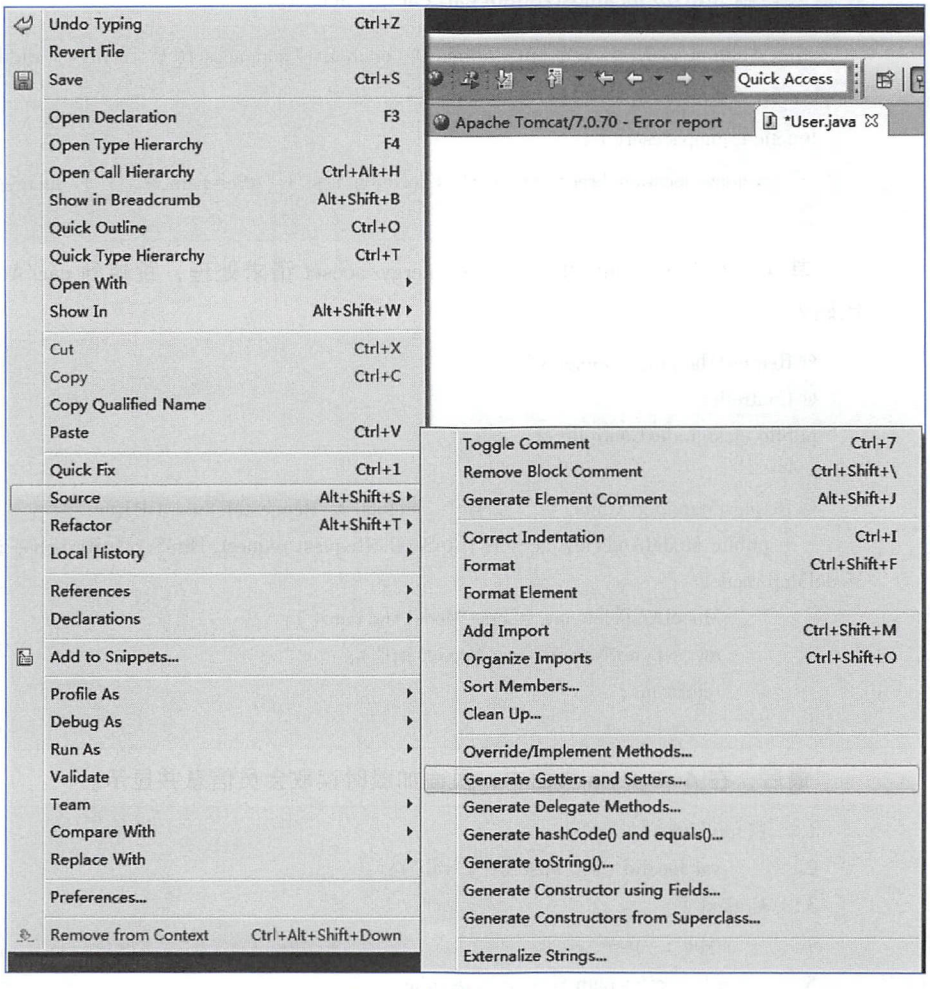


图 6-4-4　自动生成 Getters and Setters

(4) 用户信息显示的实现

① 在 IndexController 中创建方法 accset，实现请求转发。

```
@RequestMapping(value = "/accset", method = RequestMethod.GET)
public ModelAndView accset(HttpServletRequest request,HttpServletResponse response,
ModelMap model) {
    ModelAndView mv = new ModelAndView();
    mv.setViewName("car/Accset.jsp");
    return mv;
}
```

代码说明：

- 在这里控制器只是做了一个请求转发，请求来了马上转到 Accest.jsp。

② Accset.jsp 页面中用户信息的显示。

首先，在 Head.jsp 中找到查看基本信息对应的代码，单击“基本信息”按钮，使请求转到 http://localhost:8080/car/energy/accset。

```
<li><button class="btn btn-primary" onclick="jumpaccset()" style="width: 80%;
margin: 5px 19px;">基本信息</button></li>
function jumpaccset(){
    window.location.href="http://"+location.host+"<%=path%>/"+"energy/accset";
}
```

其次，在 IndexController 中定义 energy/accset 请求处理，跳转到 car/Accset.jsp，代码如下。

```
@RequestMapping("/energy")
@Controller
public class IndexController {
……
@RequestMapping(value = "/accset", method = RequestMethod.GET)
    public ModelAndView accset(HttpServletRequest request,HttpServletResponse response,
ModelMap model) {
        ModelAndView mv = new ModelAndView();
        mv.setViewName("car/Accset.jsp");
        return mv;
}
```

最后，在 Accset.jsp 页面中，页面加载时读取会员信息并显示。

```
$(function(){
    var localid=$("#userid").val();
$.ajax({
    type : "post",
    url : '<%=path%>' + "/user/accset",
    data : {
    id : localid
```

```
8.      },
9.      dataType : "jsonp",//数据类型为 json
10.     jsonp:"jsoncallback",
11.     success : function (data){
12.         var id = data.user.id;
13.         var nickname = data.user.nickname;
14.         var birthdate = data.birthdate;
15.         var sex = data.user.sex;
16.         var dltime = data.dltime;
17.         var usertype = data.user.usertype;
18.         var hpic = data.user.hpic;
19.         $("#userid").val(id);
20.         $("#nickname").val(nickname);
21.         $("input[name='sex'][value="+sex+"]").attr("checked",true);
22.         var y1 = birthdate.substring(0,4);
23.         var m1 = parseInt(birthdate.substring(5,7));
24.         var d1 = parseInt(birthdate.substring(8));
25.         $("#year").val(y1);
26.         $("#month").val(m1);
27.         $("#day").val(d1);
28.         var y2 = dltime.substring(0,4);
29.         var m2 = parseInt(dltime.substring(5,7));
30.         var d2 = parseInt(dltime.substring(8));
31.         $("#years").val(y2);
32.         $("#months").val(m2);
33.         $("#days").val(d2);
34.         if(hpic != null && !hpic.equals("")){
35.         var p = "\\"+"upload" + hpic.substring(hpic.lastIndexOf('\\'));
36.         $("#pic").attr('src',p);
37.         }
38.     }
39. });
40. });
```

代码说明：

- 第 1 行：$(function(){……})表示页面一加载就开始执行。
- 第 4 行：向"/user/accset"发送请求。
- 第 7 行~第 33 行：将返回的数据信息进行显示。

③ 在 UserController 中实现 accset 方法。

```
1.  @RequestMapping("/accset")
2.  public void accset(HttpServletRequest req,HttpServletResponse resp)
3.          throws IOException,ServletException, ParseException {
```

```
4.          User user = (User)req.getSession().getAttribute("user");
5.          SimpleDateFormat sdf=new SimpleDateFormat("yyyy-MM-dd");
6.          Date bdate = user.getBirthdate();
7.          Date dlt = user.getDltime();
8.          String birthdate = sdf.format(bdate);
9.          String dltime = sdf.format(dlt);
10.         Map<String,Object> map = new HashMap<String,Object>();
11.         map.put("user", user);
12.         map.put("birthdate", birthdate);
13.         map.put("dltime", dltime);
14.         String result=new Gson().toJson(map);
15.         String jsonp = req.getParameter("jsoncallback");
16.         resp.setCharacterEncoding("UTF-8");
17.         resp.setContentType("text/html");
18.         if(jsonp != null){
19.             result = jsonp+"("+result+")";
20.             resp.getWriter().write(result);
21.         }else{
22.             resp.getWriter().write(result);
23.         }
24.     }
```

代码说明：

- 第4行：从 sesson 中取出已经登录的账户 User 的信息。
- 第5行~第14行：将需要的信息：user、birthdate、dltime 这3个信息存入 map，并且把 map 转换为 json 格式存储在 result 变量中。
- 第15行~第23行：将 result 变量输出，这样就可以将用户信息在 Ajax 调用中获取并显示。

(5) 会员信息更新的实现

① 在 Accset.jsp 页面实现信息保存方法 save()。

```
1.  function save(){
2.      var id = $("#userid").val();
3.      var password=$("#password").val();
4.      var nickname=$("#nickname").val();
5.      var y1=$("#year").val();
6.      var m1=$("#month").val();
7.      var d1=$("#day").val();
8.      var y2=$("#years").val();
9.      var m2=$("#months").val();
10.     var d2=$("#days").val();
11.     var val=$('input:radio[name="sex"]:checked').val();
12.     var item = $('input:checkbox[name="com"]:checked').val();
```

```
13.     if(item!=1){item=0;}
14.     var birthdate = y1 + "-"+ m1 + "-"+ d1;
15.     var dltime = y2 + "-"+ m2 + "-"+ d2;
16.     $.ajax({
17.             type : "post",
18.             url : '<%=path %>' + "/user/updateuser",
19.             data : {
20.                 id : id,
21.                 password : password,
22.                 nickname : nickname,
23.                 val : val,
24.                 item : item,
25.                 birthdate : birthdate,
26.                 dltime : dltime
27.             },
28.             dataType : "jsonp",//数据类型为 json
29.             jsonp:"jsoncallback",
30.             success : function(data){
31.                 alert("保存成功!");
32.             }
33. });
34. }
```

代码说明：

- 第 2 行~第 15 行：获取信息修改页面中用户信息的参数，存放在变量中。
- 第 18 行：向"/user/updateuser"发送请求。
- 第 19 行~第 27 行：将用户信息数据定义为 data 对象。

② 在 UserController 中实现 updateUser 方法。

updateUser 方法的实现代码如下。

```
1.  package com.piesat.zyms.web.cms;
2.  @RequestMapping("/user")
3.  @Controller
4.  public class UserController {
5.  @Autowired
6.  private UserService userService;
7.  @RequestMapping("/updateuser")
8.  public void updateUser(HttpServletRequest req,HttpServletResponse resp)
        throws IOException,ServletException, ParseException {
9.          String id = req.getParameter("id");
10.         String password = req.getParameter("password");
11.         String nickname = req.getParameter("nickname");
12.         int sex = Integer.parseInt(req.getParameter("val"));
```

```
13.             int usertype = Integer.parseInt(req.getParameter("item"));
14.             String bdate = req.getParameter("birthdate");
15.             String dlt = req.getParameter("dltime");
16.             DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
17.             Date birthdate = null;
18.             Date dltime = null;
19.             try {
20.                     birthdate = format.parse(bdate);
21.                     dltime = format.parse(dlt);
22.             } catch (ParseException e) {
23.                     e.printStackTrace();
24.             }
25.             User user = new User();
26.             user.setId(id);
27.             user.setPassword(password);
28.             user.setNickname(nickname);
29.             user.setBirthdate(birthdate);
30.             user.setDltime(dltime);
31.             user.setCreatetime(new Date());
32.             user.setSex(sex);
33.             user.setUsertype(usertype);
34.             userService.updateUser(user);
35.             String result=new Gson().toJson(user);
36.             String jsonp = req.getParameter("jsoncallback");
37.             resp.setCharacterEncoding("UTF-8");
38.             resp.setContentType("text/html");
39.             if(jsonp != null){
40.                 result = jsonp+"("+result+")";
41.                 resp.getWriter().write(result);
42.             }else{
43.                 resp.getWriter().write(result);
44.             }
45.     }
46. }
```

代码说明：

- 第2行：注明url请求地址。
- 第3行：将UserController用@Controller注解，该类会自动加载到Spring容器中。
- 第5行~第6行：自动加载userService类。
- 第7行~第8行：声明updateUser方法，其请求url为user/updateuser。
- 第9行~第33行：接收参数用户参数信息，并封装在user对象中。
- 第34行：调用userService类中的方法updateUser（User user）更新用户信息。

- 第 35 行：将 user 对象转换为 json 格式存储在 result 变量中。
- 第 36 行~第 44 行：将 result 作为返回值输出给前台 Ajax 调用的页面。

③ 实现业务逻辑的处理层类 UserService. Java。

首先，在 src 文件夹中新建包 com. piesat. zyms. service. cms。

其次，在步骤①创建的包中，新建类 UserService. java，添加方法 updateUser (User User)，代码如下。

```
1.  package com. piesat. zyms. service. cms;
2.  @ Service
3.  public class UserService {
4.    @ Autowired
5.    private UserMapper userMapper;
6.    public void updateUser (User user) {
7.          userMapper. updateUser(user);
8.    }
9.  }
```

代码说明：

- 第 4 行~第 5 行：自动装载 UserMapper 对象 userMapper。
- 第 6 行~第 8 行：在 service 中通过调用 userMapper 的 updateUser 方法实现。

④ 实现对象持久化映射层 UserMapper. java、UserMapper. xml。

首先，在 src 文件中新建包 com. piesat. zyms. persistence。

其次，在上一步骤新建的包中新建接口 UserMapper. java，定义用户信息更新的方法，代码如下。

```
package com. piesat. zyms. persistence;
public interface UserMapper {
   public void updateUser (User user);
}
```

在 UserMapper. xml 文件中，增加用户信息更新方法的定义，代码如下。

```
1.  <update id="updateUser" parameterType="com. piesat. zyms. domain. core. User">
2.        update user set
3.        <if test="password!=null and password!=""">
4.              password=#{password},
5.        </if>
6.        <if test="nickname!=null and nickname!="">
7.              nickname=#{nickname},
8.        </if>
9.        <if test="birthdate!=null and birthdate!="">
10.             birthdate=#{birthdate},
11.       </if>
12.       <if test="dltime!=null and dltime!="">
```

```
13.            dltime=#{dltime},
14.        </if>
15.        <if test="hpic!=null and hpic!=''">
16.            hpic=#{hpic},
17.        </if>
18.        <if test="usertype!=null and usertype!=''">
19.            usertype=#{usertype},
20.        </if>
21.        sex=#{sex},
22.        createtime=#{createtime}
23.        where id=#{id}
24.    </update>
```

代码说明：

- 第 3 行~第 20 行：如果 password 不为空或不为"" 的话，在 SQL 语句后追加“password=#{password},”，后面的 nickname、birthdate、dltime、hpic、usertype 这些参数作类似操作。

任务 6.5　上传头像图片

【任务描述】

用户注册和信息修改时可以选择头像图片，注册或修改成功时头像图片也上传至服务器，并将图片路径保存在数据库，用户登录后可以显示会员头像图片，如图 6-5-1所示。

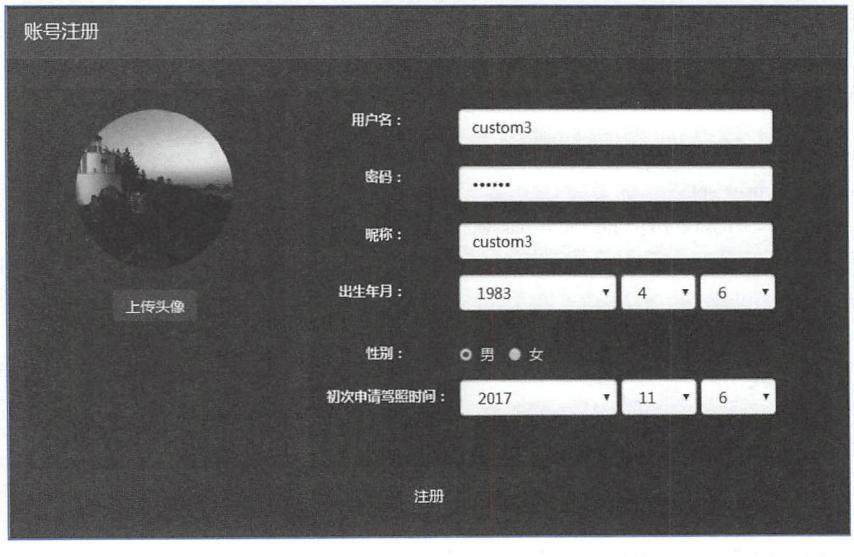
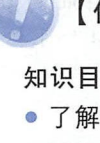


图 6-5-1　头像图片上传

【任务目标】



微课 6-6
上传头像图片

知识目标

- 了解文件上传的工作原理。
- 了解 ajaxfileupload. js 插件的使用方法。

技能目标

- 能够利用 ajaxFileUpload 插件实现图片文件的上传。

【任务分析】

上传图片过程分为以下几步：

① 下载 ajaxFileUpload 插件的 js 文件。

② 在页面中引入插件，并编写 Ajax 调用代码。

③ 实现 UserController 中的文件上传，当用户注册或者信息更新成功后执行文件上传功能。

【任务实施】

(1) 下载 ajaxfileupload. js 插件

在网络资源中 ajaxfileupload. js 可以找到很多同名的插件，本任务插件的下载地址是 https://github. com/carlcarl/AjaxFileUpload，下载界面如图 6-5-2 所示。

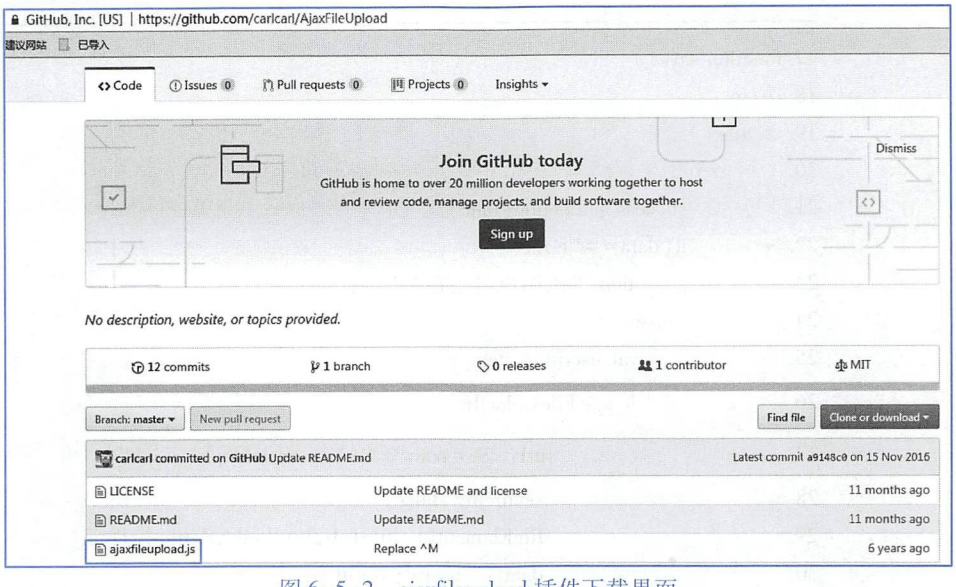


图 6-5-2　ajaxfileupload 插件下载界面

(2) 在 Registe. jsp 页面实现 Ajax 调用

① 在 Registe. jsp 页面中引入插件，代码如下。

```
<script src="resources/js/ajaxfileupload.js" type="text/javascript"></script>
```

② 单击“上传头像”按钮，触发相关事件，相关代码如下。

```
<button type="button" style="background-color:#196BA2;color:#F0F8FD;border:0px;margin-left:70px;margin-top:40px;height:30px;" class="btn btn-default">上传头像</button>

<input id="fileToUploadLink" type="file" onchange="previewFile()" style="cursor:pointer;opacity:0;margin-left:70px;margin-top:-25px;color:F0F8FD;" size="25" name="fileToUploadLink"/>
```

```
function previewFile() {
        var preview = document.getElementById("pic");
        var file    = document.querySelector('input[type=file]').files[0];
        var reader = new FileReader();
        reader.onloadend = function () {
          preview.src = reader.result;
        }
        if (file) {
          reader.readAsDataURL(file);
        } else {
          preview.src = "";
        }
}
```

```
function save(){
……
$.ajax({
    ……
        success : function(data){
        if(data==="error"){
            alert("该用户名已存在!");
        }else{
            var userid = data.id
            $.ajaxFileUpload({
                    url:'<%=path %>'+"/user/adduploadLink",
                    secureuri :false,
                    fileElementId :'fileToUploadLink',//file 控件 id
                    data : {userid:userid},
                    success : function (data){
                        alert("注册成功!");
                window.location.href="http://"+location.host+"<%=path%>/"+"energy/login";
                    },
            });
        }
```

```
36. }
37.  });
38. }
```

$.ajaxFileUpload([options])语法说明：

- url：上传处理程序地址。
- fileElementId：需要上传的文件域的 ID，即<input type=" file" >的 ID。
- secureuri：是否启用安全提交，默认为 false。
- dataType：服务器返回的数据类型。可以为 xml，script，json，html。如果不填写，jQuery 会自动判断。
- success：提交成功后自动执行的处理函数，参数 data 就是服务器返回的数据。
- error：提交失败自动执行的处理函数。
- data：自定义参数。比较有用，当有数据是与上传的图像相关时，使用该参数。
- type：当要提交自定义参数时，该参数要设置成 post。

(3) UserController 中实现"/user/adduploadLink"

在 UserController 中新建方法，代码如下。

```
1.  @ResponseBody
2.  @RequestMapping(value="/adduploadLink",method={RequestMethod.POST})
3.  public void adduploadLink(@RequestParam("fileToUploadLink")MultipartFile file,
4.          HttpServletRequest request,HttpServletResponse response,ModelMap model)
    throws ServletException {
5.  try {
6.      String userid = request.getParameter("userid");
7.      User user = userService.getUserById(userid);
8.      String originalFilename = file.getOriginalFilename();
9.      //默认图片
10.     String hpic = "";
11.     if(originalFilename!=null&&! originalFilename.equals("")){
12.         String uploadName = UUID.randomUUID().toString();
13.         String extend = originalFilename.substring(originalFilename.lastIndexOf(".")
    + 1);
14.         //生成文件相对地址
15.         hpic = "\\car\\upload1\\" + uploadName +'.'+extend;
16.         user.setHpic(hpic);
17.         //获取文件后缀
18.         String path = request.getSession().getServletContext().getRealPath
    ("/upload1")+ separator + uploadName +'.'+extend;
19.         String pathFile = request.getSession().getServletContext().getRealPath
    ("/upload1");
20.         File dirPath = new File(pathFile);
21.         File uploadFile = new File(pathFile + separator + uploadName+'.'+extend);
```

```
22.             if (!dirPath.exists()) {
23.                 uploadFile.mkdirs();
24.             }
25.             try{
26.                 //上传文件上传到服务器
27.                 file.transferTo(uploadFile);
28.             }catch(Exception e){
29.                 System.out.println(e.getMessage());
30.             }
31.         }
32.         try{
33.             userService.updateUser(user);
34.             if(user.getHpic()==null||user.getHpic().equals(""))
35.                 user.setHpic("\\car\\resources\\login\\nlogin.jpg");
36.             request.getSession().setAttribute("user", user);
37.         }catch(Exception e){
38.             e.printStackTrace();
39.         }
40.     } catch (Exception e) {
41.         e.printStackTrace();
42.     }
43. }
```

代码说明：

- 第 3 行：参数 MultipartFile file 代表选择的头像文件。
- 第 8 行：获取文件的名称。
- 第 12 行：随机生成文件名称。
- 第 13 行：获得上传文件的后缀名。
- 第 18 行~第 19 行：生成文件上传的物理路径。
- 第 20 行~第 30 行：将文件上传到物理路径，如果文件夹不存在则创建文件夹。
- 第 34 行~第 35 行：如果没有选择上传的头像文件，则数据库中为 null，但是显示的 User 对象中头像图片路径设置为\\car\\resources\\login\\nlogin.jpg。

技能训练

1. 基于 SSM 框架实现会员登录

任务描述

基于 Spring+SpringMVC+MyBatis 框架并结合 Ajax 技术实现用户登录，要求输入用户名和密码，判断登录是否成功。

任务分析

用户输入用户名和密码，根据用户名查找用户是否存在，如果用户存在且密码

匹配则返回 true，并将 user 放入 session，否则返回 false。

2. 基于 SSM 框架实现会员注册

任务描述

基于 Spring+SpringMVC+MyBatis 框架并结合 Ajax 技术实现用户注册，要求将用户信息添加到数据库，并将头像图片上传。

任务分析

用户输入一系列个人信息，单击“保存”按钮，将用户信息添加到数据库，当用户选择头像图片时，将头像图片上传，并更新数据库 user 表中 hpic 字段信息。

项目总结

在本项目中通过对个人会员信息管理模块的实现，重点阐述了以下几项任务技能。

- Bootstrap 框架的下载、配置及开发。
- 通过 jQuery 实现 Ajax 技术。
- Spring+SpringMVC+MyBatis 框架的下载、配置整合。
- 基于 Spring+SpringMVC+MyBatis 框架的功能开发。
- 会员头像图片上传。

以上这些技术基本涵盖了整个系统开发所用到的技术，但本项目业务逻辑比较简单，只涉及到会员信息的简单操作，因此还需要在后续模块的训练中继续强化基于 SSM 的框架编程技能。

项目 7　车辆信息管理模块

PPT 车辆信息管理模块

项目描述

本项目通过车辆信息管理模块的开发，详细讲解 SpringMVC+MyBatis 框架的综合应用，具体功能包括会员注册、登录、信息显示、修改等功能。

知识目标

- 熟悉 SpringMVC+MyBatis 框架的开发方法。
- 熟悉 EL+JSTL 显示方式。
- 熟悉分页控件的使用方法。

技能目标

- 能熟练运用 SpringMVC+MyBatis 开发框架。
- 能使用 EL+JSTL 实现 View 层数据的显示。
- 能熟练使用分页控件实现数据的分页显示。

任务列表

任务编号	任务名称	建议课时
任务 7.1	新增车辆信息	4
任务 7.2	显示车辆信息	1
任务 7.3	分页显示车辆信息	3
任务 7.4	显示车辆信息详情	4
任务 7.5	更新车辆信息	2
任务 7.6	解除绑定个人车辆信息	2
技能训练	查询车辆信息 删除车辆信息	2 2
	总计：	20 课时

任务 7.1　新增车辆信息

【任务描述】

通过 Bootstrap+SpringMVC+MyBatis 框架实现车辆信息新增页面，效果如图 7-1-1 所示。

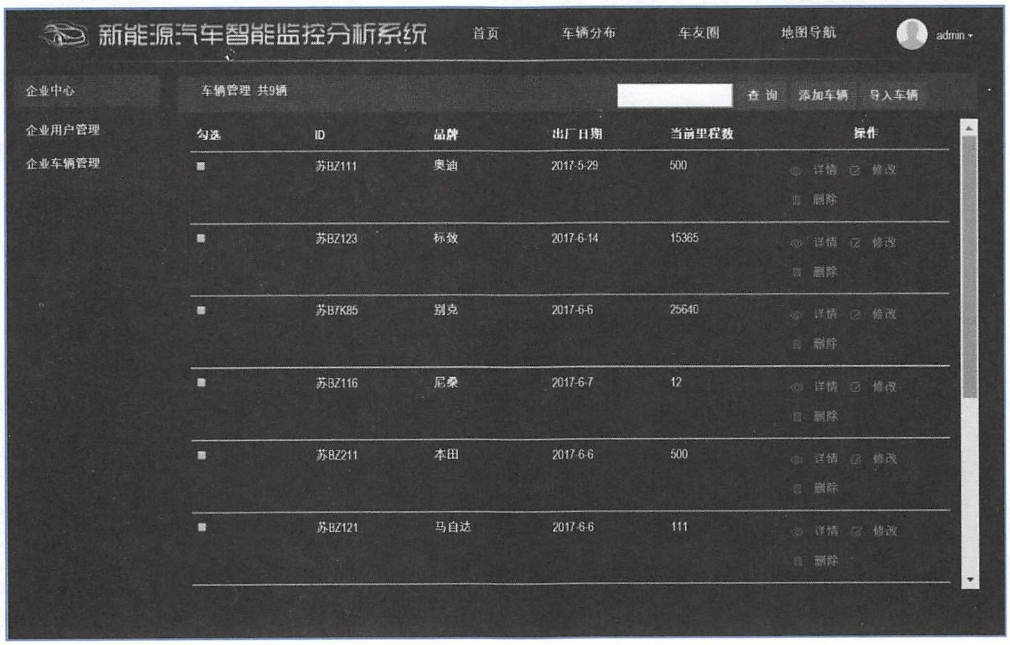


图 7-1-1　添加车辆信息页面效果

【任务目标】

微课 7-1
我的车辆信息页展示

知识目标

- 熟悉 MyBatis 框架架构和原理。
- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法。

技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据添加。

【任务分析】

采用 Bootstrap 可以快速实现网页前台，利用 Bootstrap 中的表单相关样式实现登录、注册的页面效果。

添加车辆信息的流程为：单击“添加车辆”按钮，弹出添加车辆页面，填写好车辆相关信息后，单击“提交”按钮将数据更新到后台数据库。SpringMVC 的相关配置已经在项目 6 中详细说明，本任务中不再赘述，具体的开发流程如下：

① MyBatis 配置，包括 jdbc. properties、applicationContext. xml、SqlMapConfig. xml。

② 创建汽车信息数据对应 POJO 类 CarMessage. java，定义数据结构，其和表结构一致。

③ 添加汽车的实现：

- 创建视图层 manage. jsp 页面，在页面中设计添加车辆层 modal。
- 在视图层 manage. jsp 页面，编写 Ajax 方法实现将数据推送到控制层。
- 在 CarMessageController 中实现 add 方法，用于数据交互。
- 创建业务逻辑的处理层类 CarMessageService. Java，声明 getCarMessageByUser、save 方法。
- 实现对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 queryNumberById (vehId)、queryNumberByplateNumber (plateNumber)、save( )、CarMessage getCarMessageByUser(userid)。
- 新建 CarMessageController 类，并在类中添加 add( ) 实现方法，用于将请求转发至 manage. jsp。

【任务实施】

(1) 引入 CSS 和基础 JavaScript 文件

在“<head></head>”标签区域，引入相关 CSS 样式文件。

```
<link rel="stylesheet"
    href="resources/css/bootstrap-table/bootstrap-table.css" />
<link rel="stylesheet" href="resources/css/sweetalert.css" />
<link rel="stylesheet"
    href="resources/css/bootstrap-datetimepicker.min.css" />
```

在“<body></body>”标签区域，引入相关 js 文件，需要说明的是引用 Bootstrap 的 js 文件，必须同时引用 jQuery 的 js 文件。

```
<script src="resources/js/jquery-1.9.1.min.js" type="text/javascript"></script>
<script src="res/bootstrap/js/bootstrap.min.js" type="text/javascript"></script>
```

(2) 添加车辆信息

添加“添加车辆”基本布局层 modal，默认设置该层“隐藏”。

```
<div id="modal" class="modal mymodal" aria-hidden="true"
style="display: none;left:10%" >
<div class="modal-dialog ">
    <div class="modal-content">
        <div class="modal-header">
            <button type="button" class="close" data-dismiss="modal"
                aria-hidden="true">×</button>
            <h4 class="modal-title">车辆信息</h4>
        </div>
        <div class="modal-body">
```

```
                <form id="user-create-form" class="form-horizontal" method="post"
                    action="/admin/user/create" novalidate="novalidate">
                    <div class="row form-group">
                    <div class="col-md-3 control-label">
                    <label for="vehID">车辆 ID<span class="xb">*</span></label>
                    </div>
                <div class="col-md-7 controls">
                    <input type="text" id="vehID" name="vehID" class="form-control">
                    <div class="help-block" style="display:none;"></div>
                </div>
            </div>
            <div class="row form-group">
                <div class="col-md-3 control-label">
                    <label for="plateNumber">车牌号<span class="xb">*</span>
</label>
                </div>
                <div class="col-md-7 controls">
                    <input type="text" id="plateNumber" name="plateNumber"
                                                class="form-control">
                    <div class="help-block" style="display:none;"></div>
                </div>
            </div>
            <div class="row form-group">
                <div class="col-md-3 control-label">
                <label for="brand">品牌<span class="xb">*</span></label>
                </div>
            <div class="col-md-7 controls">
                <input type="text" id="brand" name="brand" class="form-control">
                <div class="help-block" style="display:none;"></div>
            </div>
            </div>
            <div class="row form-group">
                <div class="col-md-3 control-label">
                    <label for="currentMileage">当前里程数</label>
                </div>
                <div class="col-md-7 controls">
                <input type="text" id="currentMileage" name="currentMileage"
                                                class="form-control">
                <div class="help-block" style="display:none;"></div>
            </div>
                    </div>
                    <div class="row form-group">
                    <label for="produceDate" class="col-md-3 control-label">出厂日期</label>
```

```
                    <div class="col-md-7 controls">
                    <input type="text" class="form-control Wdate" id="produceDate"
placeholder="请选择开始时间" readonly="true"
    onfocus="WdatePicker({dateFmt:'yyyy-MM-dd'});" />
                                    </div>
                                </div>
                                <div class="row form-group">
                                    <div class="col-md-3 control-label">
                                        <label for="userId">归属人</label>
                                    </div>
                                    <div class="col-md-7 controls">
                                        <select id="userId">
    <c:forEach var="user" items="${users}" varStatus="vs">
                    <option value="${user.id}">${user.username}</option>
                                            </c:forEach>
                                        </select>
<div class="help-block" style="display:none;"></div>
                                    </div>
                                </div>
                                <div class="row form-group">
                                    <div class="col-md-3 control-label">
                                        <label for="remarks">说明</label>
                                    </div>
                                    <div class="col-md-7 controls">
                                        <input type="text" id="remarks" name="remarks"
                                            class="form-control">
                                        <div class="help-block" style="display:none;"></div>
                                    </div>
                                </div>
                            </form>
                        </div>
                        <div class="modal-footer">
                            <button type="submit" class="btn btn-primary pull-right"
                id="addsure" style="margin-left: 5px">提交</button>
                            <button type="button " class=" btn btn-primary pull-right"
                                data-dismiss="modal" aria-hidden="true">取消</button>
                        </div>
                    </div>
                </div>
            </div>
```

(3) 为“添加车辆”按钮设置属性

在/WebContent/WEB-INF/page/ace/car 文件夹下新建用户管理页面文件 manage.jsp。设置属性 data-toggle 为 modal，当用户单击该按钮时，将打开“添加车辆”

对话框，如图 7-1-2 所示。

```
<a class="btn btn-primary btn-sm" data-toggle="modal"
                                    data-target="#modal">添加车辆</a>
```

图 7-1-2　添加车辆信息

(4) 为提交按钮添加单击事件

① 在 modal 层中找到“提交”按钮相关代码，在代码中添加单击事件代码，具体代码如下。

```
<button type="submit" class="btn btn-primary pull-right"
    id="addsure"  onclick="addCar()"  style="margin-left: 5px">提交
</button>
```

② 定义函数 addCar()。在<body></body>中定义 login 函数。

```
1.  function addCar(){
2.              var vehID=$("#vehID").val();
3.              var plateNumber = $("#plateNumber").val();
4.              var brand = $("#brand").val();
5.              var remarks = $("#remarks").val();
6.              var produceDate = $("#produceDate").val();
7.              var currentMileage =parseInt( $("#currentMileage").val());
8.              var userId = $("#userId option:selected").val();
9.              $.ajax({
10.             type:"post",
11.             url:'<%=path%>'+"/carmessage/add",
12.             dataType : "jsonp",
13.             jsonp:"jsoncallback",
14.             data:{
15.             "vehID":vehID,
16.             "plateNumber":plateNumber,
```

```
17.                 "brand":brand,
18.                 "remarks":remarks,
19.                 "produceDate":produceDate,
20.                 "currentMileage":currentMileage,
21.                 "userId":userId,
22.                 },
23.                 success : function(data){
24.                     swal("", "添加成功", "success");
25.                         $("#modal").modal("hide");
26.                                 window.location.href="http://" + location.host +
    "<%=path%>/" + "carmessage/all";
27.                     }
28.                 });
29.             }
```

代码说明：

- 第 2 行~第 8 行：定义变量分别获取并存储车辆 ID、车牌号、品牌、当前里程数、出厂日期、归属人、说明等相关信息，$("#userId option:selected").val()表示获取用户选择项。
- 第 15 行~第 21 行：将 vehID、plateNumber、brand、remarks、produceDate、currentMileage 和 userId 作为参数进行传递。
- 第 9 行~第 28 行：$ajax()方法调用需要设置很多参数，其中 type 表示请求方式为 post，url 为请求地址，data 传输数据参数的值就是在第 2~8 行所定义的变量，dataType 表示数据类型为 jsonp 格式，success 表示请求成功后调用的回调函数，如果成功则提示“添加成功!”，并将添加车辆层隐藏，并跳转到车辆显示页面。

（5）创建车辆表对应的 POJO 类 CarMessage. java

CarMessage 类是对 CarMessage 表的一个映射，类的属性与表的字段一一对应，如图 7-1-3 所示。

名	类型	长度
id	int	11
vehID	varchar	20
plateNumber	varchar	50
brand	varchar	255
remarks	varchar	255
producedate	date	0
currentmileage	int	255
userid	varchar	255
createid	varchar	255
cartype	varchar	255

(a)

```
public class CarMessage {
    private Integer id;
    private String vehID;
    private String plateNumber;
    private String brand;
    private String remarks;
    private Date produceDate;
    private Integer currentMileage;
    private String userId;
    private String createId;
    private String cartype;
    private int count;
    private Distribution distribution;
```

(b)

图 7-1-3　CarMessage 表字段与 CarMessage 类属性对应图

① 在 src 包中，在包 com. piesat. zyms. domain. core 中新建一个类 CarMessage，类的属性如图 7-1-3 所示。

② 创建好属性后，另起一行，右击，在弹出的快捷菜单中选择“Source→Generate Getters and Setters”菜单命令，如图 7-1-4 所示。

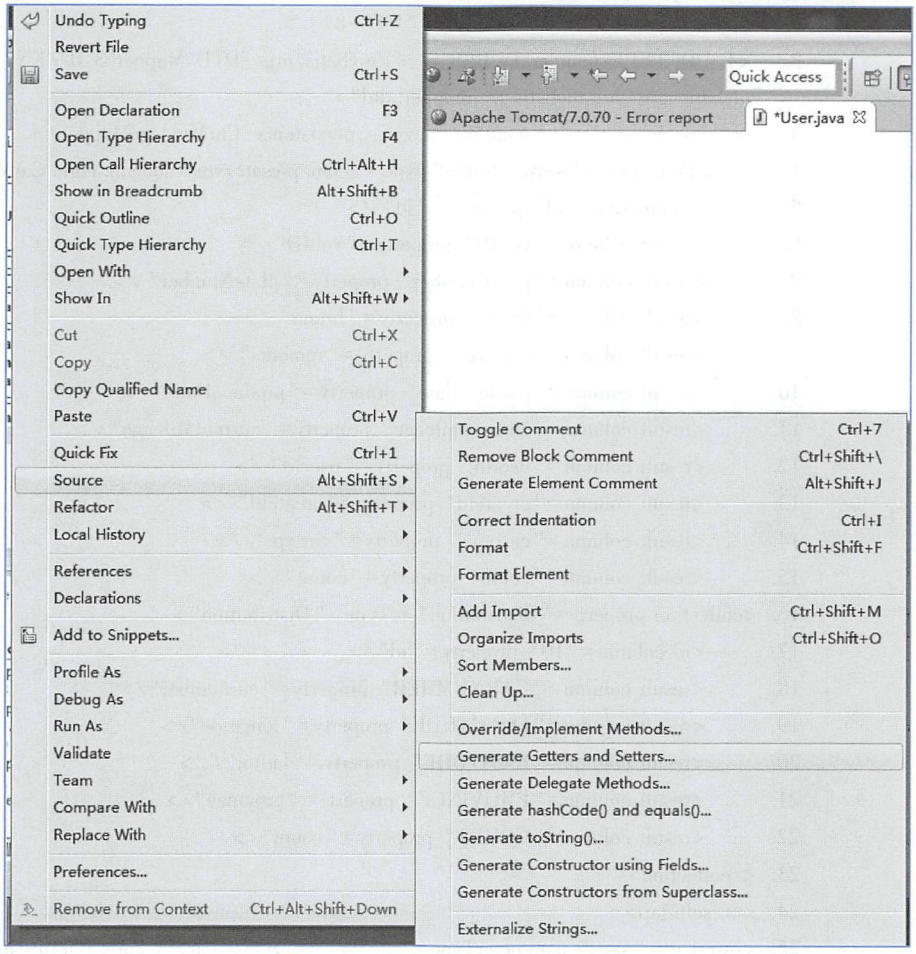


图 7-1-4　自动生成 getters and setters

(6) 实现对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml

① 在 src 文件包 com. piesat. zyms. persistence 中新建接口 CarMessageMapper. java，定义 save 方法，实现车辆的添加。定义 queryNumberById 方法，根据车牌识别 ID 查询车辆数，判断车辆是否已经存在。定义 queryNumberByplateNumber 方法，实现根据品牌查询记录数。定义根据 getCarMessageByUser 方法，实现根据 userid 获取车辆，代码如下。

```
package com. piesat. zyms. persistence;
public interface CarMessageMapper{
    public void save(CarMessage carMessage);
    public long queryNumberById(String vehId);
    public long queryNumberByplateNumber(String plateNumber);
    public CarMessage getCarMessageByUser(String userid);
}
```

② 继续在步骤①创建的包中新建文件 CarMessageMapper. xml，用于映射数据表 CarMessage 的字段和表上的操作，代码如下。

```
<? xml version = " 1. 0" encoding = " UTF-8" ? >
<! DOCTYPE mapper PUBLIC " -//mybatis. org//DTD Mapper 3. 0//EN" " http://mybatis. org/dtd/mybatis-3-mapper. dtd" >
<mapper namespace = " com. piesat. zyms. persistence. CarMessageMapper" >
<resultMap id = " BaseResultMap" type = " com. piesat. zyms. domain. core. CarMessage" >
    <id column = " id" property = " id" />
    <result column = " vehID" property = " vehID" />
    <result column = " plateNumber" property = " plateNumber" />
    <result column = " brand" property = " brand" />
    <result column = " remarks" property = " remarks" />
    <result column = " producedate" property = " produceDate" />
    <result column = " currentmileage" property = " currentMileage" />
    <result column = " userid" property = " userId" />
    <result column = " createid" property = " createId" />
    <result column = " cartype" property = " cartype" />
    <result column = " count" property = " count" />
<collection property = " distribution" ofType = " Distribution" >
    <id column = " ID" property = " id" />
    <result column = " CARNUMBER" property = " carnumber" />
    <result column = " LONGTUDE" property = " longtude" />
    <result column = " LATITUDE" property = " latitude" />
    <result column = " PROVINCE" property = " province" />
    <result column = " COUNT" property = " count" />
</collection>
</resultMap>
<insert id = " save" parameterType = " com. piesat. zyms. domain. core. CarMessage" >
    insert into
carmessage( vehID, plateNumber, brand, remarks, producedate, currentmileage, userid, createid)
values( #{ vehID} , #{ plateNumber} , #{ brand} , #{ remarks} , #{ produceDate} , #{ currentMileage} , #{ userId} , #{ createId} )
</insert>
<select id = " queryNumberById" resultType = " java. lang. Long" >
    select count( * ) from
    carmessage where vehId = #{ vehId}
</select>
<select id = " queryNumberByplateNumber" resultType = " java. lang. Long" >
    select count( * ) from
    carmessage where plateNumber = #{ plateNumber}
</select>
```

```
38.  <select id="getCarMessageByUser" parameterType="java.lang.String" resultType
     ="com.piesat.zyms.domain.core.CarMessage">
39.      select * from carmessage where userid = #{userid}
40.  </select>
41. </mapper>
```

代码说明：

- 第 2 行：声明是 MyBatis 的文件配置。
- 第 3 行：声明命名空间，CarMessageMapper.java 接口中声明的方法，将在 CarMessageMapper.xml 文件中给出详细的实现。
- 第 4 行~第 15 行：将数据表 carmessage 的字段与 POJO 类 CarMessage 的属性进行映射，注意，如果表中的字段和类的属性名称完全一致（包括大小写都一致），此部分代码也可以省略，默认匹配，例如在这里的第 4 行~第 15 行也可以省略。
- 第 25 行~第 29 行：定义用户添加的方法 save（），因为是数据库添加，所以标签用<insert></insert>，输入参数是 CarMessage，无返回值。
- 第 30 行~第 33 行：定义根据车辆识别号查询车辆数方法 queryNumberById()（可根据查得的结果判断该用户是否已经绑定车辆），因为是数据库查询，所以标签用<select></select>，输入参数是 vehId，返回记录数。
- 第 34 行~第 37 行：定义车辆品牌信息查询车辆数的方法 queryNumberByplateNumber()，因为是数据库查询，所以标签用<select></select>，输入参数是 plateNumber，返回记录数。
- 第 38 行~第 40 行：定义根据用户 ID 查询汽车信息的方法 getCarMessageByUser()，因为是数据库查询，所以标签用<select></select>，输入参数是 userid，返回汽车信息。

（7）实现业务逻辑的处理层类 CarMessageService.Java

在 src 文件夹包 com.piesat.zyms.service.cms 中，新建类 CarMessageService.java，代码如下。

```
1.  package com.piesat.zyms.service.cms;
2.  @Service
3.  public class CarMessageService {
4.      @Autowired
5.      private CarMessageMapper carMessageMapper;
6.      public CarMessage save(CarMessage carMessage) {
7.          carMessageMapper.save(carMessage);
8.          return carMessage;
9.      }
10.     public CarMessage getCarMessageByUser(String userid) {
11.         return carMessageMapper.getCarMessageByUser(userid);
12.     }
13. }
```

代码说明：

- 第 2 行：如果一个类带了@ Service 注解，将自动注册到 Spring 容器，不需要再在配置文件定义 bean 了，但需要在配置文件中增加扫描文件包，例如在 cms-servlet-admin. xml 中的如下代码。

```
<! --把标记了@ Controller 注解的类转换为 bean -->
<context:component-scan base-package="com. piesat. zyms. service" />
```

@ Controller 也是一样的道理，在配置文件 cms-servlet-admin. xml 中也有类似代码。

```
<! --使用组件扫描的方式可以一次扫描多个 Controller -->
<context:component-scan base-package="com. piesat. zyms. web" />
```

- 第 4 行~第 5 行：自动装载 CarMessageMapper 对象 carMessageMapper。
- 第 6 行~第 9 行：在 service 中通过调用 CarMessageMapper 的 save 方法实现。
- 第 10 行~第 12 行：在 service 中通过调用 CarMessageMapper 的 getCarMessageByUser 方法实现。

(8) 实现控制层 CarMessageController. java

在 com. piesat. zyms. web. cms 包中，新建 Java 类 CarMessageController，代码如下。

```
1.  package com. piesat. zyms. web. cms;
2.  @ Controller
3.  @ RequestMapping("/carmessage")
4.  public class CarMessageController {
5.  @ Autowired
6.  private CarMessageService carMessageService;
7.  @ Autowired
8.  private UserService userService;
9.  @ RequestMapping("/add")
10. public void addCar(HttpServletRequest req, HttpServletResponse resp)
11.         throws IOException {
12.     // TODO 获取当前登录用户的 ID
13.     HttpSession session=req. getSession();
14.     String userId = req. getParameter("userId");
15.     String createId=userId;
16.     String vehID = req. getParameter("vehID");
17.     String plateNumber = req. getParameter("plateNumber");
18.     long total = carMessageService. queryCountByid(vehID);
19.     long total_ = carMessageService. queryCountByNumber(plateNumber);
20.     CarMessage cm = carMessageService. getCarMessageByUser(userId);
21.     String result = null;
22.     if (cm == null) {
23.         if (createId == null || total > 0 || total_ > 0) {
24.             result = new Gson(). toJson("error");
25.         } else {
```

```
26.                 String brand = req.getParameter("brand");
27.         Integer currentMileage = Integer.parseInt(req.getParameter("currentMileage"));
28.                 String date = req.getParameter("produceDate");
29.                 Date produceDate = null;
30.                 DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
31.                 try {
32.                     produceDate = format.parse(date);
33.                 } catch (Exception e) {
34.                 }
35.                 String remarks = req.getParameter("remarks");
36.                 CarMessage carMessage = new CarMessage();
37.                 carMessage.setVehID(vehID);
38.                 carMessage.setBrand(brand);
39.                 carMessage.setCurrentMileage(currentMileage);
40.                 carMessage.setCreateId(createId);
41.                 carMessage.setPlateNumber(plateNumber);
42.                 carMessage.setProduceDate(produceDate);
43.                 carMessage.setRemarks(remarks);
44.                 carMessage.setUserId(userId);
45.                 carMessageService.save(carMessage);
46.                 result = new Gson().toJson(carMessage);
47.             }
48.             String jsonp = req.getParameter("jsoncallback");
49.             resp.setCharacterEncoding("UTF-8");
50.             resp.setContentType("text/html");
51.             if (jsonp != null) {
52.                 result = jsonp + "(" + result + ")";
53.                 resp.getWriter().write(result);
54.             } else {
55.                 resp.getWriter().write(result);
56.             }
57.         } else {
58.             result = new Gson().toJson("false");
59.             String jsonp = req.getParameter("jsoncallback");
60.             resp.setCharacterEncoding("UTF-8");
61.             resp.setContentType("text/html");
62.             if (jsonp != null) {
63.                 result = jsonp + "(" + result + ")";
64.                 resp.getWriter().write(result);
65.             } else {
66.                 resp.getWriter().write(result);
67.             }
68.         }}
```

代码说明：

- 第 3 行：注明 url 请求地址。
- 第 2 行：将 CarMessageController 用@ Controller 注解，该类会自动加载到 Spring 容器中。
- 第 5 行~第 8 行：自动加载 carMessageService，userService 类。
- 第 10 行~第 11 行：声明 addCar 方法，其请求 url 为/carmessage/addCar。
- 第 13 行~第 14 行：接收参数用户编号 userId。
- 第 15 行~第 16 行：接收参数车辆编号 vehID、车辆车牌号 plateNumber。
- 第 17 行~第 18 行：分别根据车辆编号和车牌编号查询已有车辆数。
- 第 19 行：接收参数信息，获取归属人的用户 ID。
- 第 20 行：调用 CarMessageService 类的 getCarMessageByUser 方法，将结果存放在变量 cm 中。
- 第 21 行~第 25 行：如果用户输入的车牌号和品牌信息已经存在或选择的用户名下已经有车辆则返回出错信息。
- 第 26 行~第 47 行：将用户输入的车辆信息封装到 CarMessage 对象，并将对象转换为 json 格式存储在 result 变量中。
- 第 48 行~第 56 行：将 result 作为返回值输出给前台 Ajax 调用的页面。
- 第 57 行~第 67 行：如果输入的信息不符合要求，将出错信息输出给前台 Ajax 调用的页面。

任务 7.2　显示车辆信息

【任务描述】

实现车辆信息显示，效果如图 7-2-1 所示。

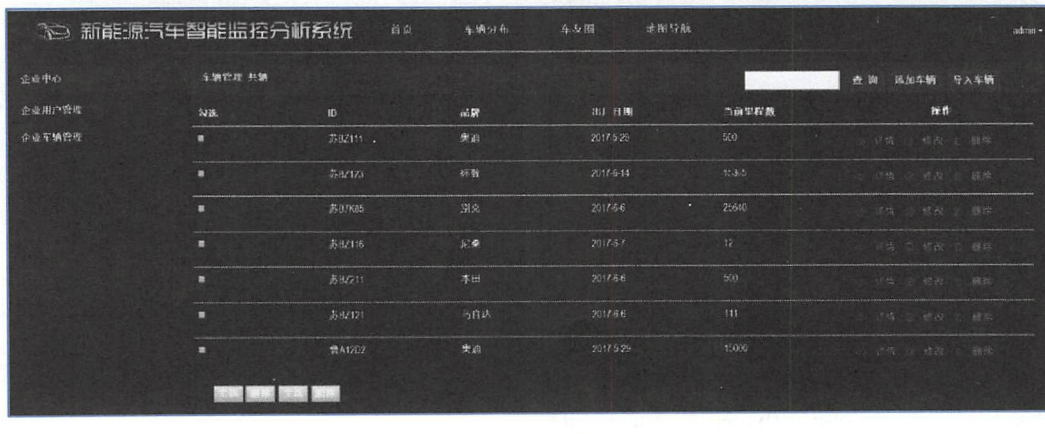


图 7-2-1　车辆信息显示

【任务目标】

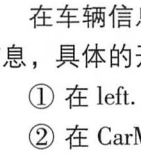
知识目标

- 熟悉 MyBatis 框架架构和原理。
- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法。

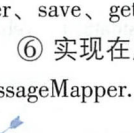
技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据显示。

微课 7-2
个人车辆绑定



微课 7-3
个人车辆详情展示和删除绑定



微课 7-4
企业用户管理

【任务分析】

在车辆信息显示页面，当用户单击“企业车辆管理”超链接时，则显示所有的信息，具体的开发流程如下。

① 在 left. jsp 页面的“企业车辆管理”链接，将请求发送至 Controller 层。

② 在 CarMessageController 中实现 showCar 方法，并将结果返回 manage. jsp。

③ 在 manage. jsp 页面向 CarMessageController 层发送获取汽车信息请求。

④ 实现 CarMessageController 中的 search( ) 方法。

⑤ 在业务逻辑的处理层类 CarMessageService. Java 文件中声明 getCarMessageByUser、save、getAllCarMessage 方法。

⑥ 实现在服务层中需要调用的对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 getAllCarMessage( )。

【任务实施】

① 在 left. jsp 页面设置“企业车辆管理”超链接的 href 属性为/car/carmessage/all，代码如下。

```
1.  <div class="d">
2.      <a class="font" id="safe" href="/car/carmessage/all">企业车辆管理</a>
3.  </div>
```

② 在 CarMessageController 中实现 showCar 方法，代码如下：

```
1.  @RequestMapping("/all")
2.  public ModelAndView showCar(HttpServletRequest request,
3.          HttpServletResponse response, ModelMap model) {
4.      ModelAndView mv = new ModelAndView();
5.      mv.setViewName("car/manage.jsp");
6.      List<User> users = userService.getUser();
7.      mv.addObject("users", users);
8.      return mv;
9.  }
```

代码说明：

- 调用 userService 服务的 getUser( ) 方法，获取用户记录，并封装到 ModelAndView 模型中，并请求转到 car/manage. jsp 页面。

③ 在 manage. jsp 页面中显示列表。在页面中添加页面加载代码如下。

```
<script type="text/javascript">
    //初始化页面
    $(function(){
        search();
    })
    //显示列表
    function search(){
    $.ajax({
        type:"post",
        url:'<%=path%>'+"/carmessage/search1",
        dataType : "jsonp",
        jsonp:"jsoncallback",
        success : function(data){
            var listdata=data.rows;
            var html="" ;
            for(var f = 0; f < ((!! listdata) ? listdata.length : 0); f++) {
                html+="<tr>";
                html+="<td ><input type=\"checkbox\" id=\""+listdata[f].vehID
+"\" name=\"box\" /></td>";
                html+="<td >"+listdata[f].vehID+"</td>";
                html+="<td >"+listdata[f].brand+"</td>";
                var now=listdata[f].produceDate;
                  if(produceDate!=0){
                  now_=formatDate(now);
                html+="<td >"+now_+"</td>";
                  }else{
                  html+="<td ></td>";
                  }
            html+="<td >"+listdata[f].currentMileage+"</td>";
            html+="<td style='width: 22%;'><div style='float: left;'><img alt='del'
src='resources/images/view.png'><a class=\"btn btn-link editdeptbtn\" style='color: #
20c8ff' onclick=\"view('"+listdata[f].vehID+"');\">详情</a></div>";
            html+="<div style='float: left;'><img alt='del' src='resources/images/
edit.png'><a class=\"btn btn-link editdeptbtn\" data-toggle=\"modal\" data-target=
\"#modalupdate\" style='color: #20c8ff' onclick=\"javascript:carupdate("+listdata[f]
.vehID+");\">修改</a></div>";
            html+="<div style='float: left;'><img alt='del' src='resources/images/de-
lete.png'><a class=\"btn btn-link editdeptbtn\" style='color: #20c8ff' data-toggle=\"modal
\" onclick=\"javascript:cardelete("+listdata[f].vehID+");\">删除</a></div>";
            html+="</tr>"
        }
        $("#tablelist").html(html);
```

```
35.                    }});}
36. </script>
```

代码说明：

- 第 2 行~第 5 行：初始化页面，调用 search（）方法。
- 第 7 行~第 35 行：触发 Ajax 方法，发送查询请求 carmessage/search，执行成功后以 JSON 格式将数据返回，同时将数据输出到页面。
- 第 14 行：Ajax 执行成功后，返回数据 data，获取列表记录数存放在 listdata 中。
- 第 16 行~第 33 行：将 data 中的数据逐条读出，并存放于 html 中。
- 第 34 行：将读取出的数据在表格中显示。

④ 实现 CarMessageController 中的 search( )方法。在步骤 3 的第 10 行代码中，将查询列表的请求发送到控制层中，因此需要为其添加 search( )方法的实现，代码如下。

```
1.  @RequestMapping("/search")
2.  public void search(HttpServletRequest req, HttpServletResponse resp) throws
    IOException {
3.      List<CarMessage> carMessages = carMessageService.getAllCarMessage();
4.      List<Carshow> carshows = new ArrayList<>();
5.      if (carMessages.size() > 0 && carMessages != null) {
6.          for (CarMessage carMessage : carMessages) {
7.              Carshow carshow = new Carshow();
8.              carshow.setVehID(carMessage.getVehID());
9.              carshow.setCurrentMileage(carMessage.getCurrentMileage());
10.             if (carMessage.getProduceDate() != null)
11.                 carshow.setProduceDate(carMessage.getProduceDate()
12.                         .getTime());
13.             carshow.setBrand(carMessage.getBrand());
14.             carshows.add(carshow);
15.         }
16.     }
17.     TableReturnObject tableReturnObject = new TableReturnObject();
18.     tableReturnObject.setTotal(carshows.size());
19.     tableReturnObject.setRows(carshows);
20.     String result = new Gson().toJson(tableReturnObject);
21.     String jsonp = req.getParameter("jsoncallback");
22.     resp.setCharacterEncoding("UTF-8");
23.     resp.setContentType("text/html");
24.     if (jsonp != null) {
25.         result = jsonp + "(" + result + ")";
26.         resp.getWriter().write(result);
27.     } else {
```

```
28.             resp.getWriter().write(result);
29.         }
30.     }
31. public static class Carshow {
32.         private String vehID;
33.         private String brand;
34.         private long produceDate;
35.         private Integer currentMileage;
36.         public String getVehID() {
37.             return vehID;
38.         }
39.         public void setVehID(String vehID) {
40.             this.vehID = vehID;
41.         }
42.         public String getBrand() {
43.             return brand;
44.         }
45.         public void setBrand(String brand) {
46.             this.brand = brand;
47.         }
48.
49.         public long getProduceDate() {
50.             return produceDate;
51.         }
52.         public void setProduceDate(long produceDate) {
53.             this.produceDate = produceDate;
54.         }
55.         public Integer getCurrentMileage() {
56.             return currentMileage;
57.         }
58.
59.         public void setCurrentMileage(Integer currentMileage) {
60.             this.currentMileage = currentMileage;
61.         }
62.     }
63.  public static class TableReturnObject {
64.         private long total;
65.         private List<? > rows;
66.         public long getTotal() {
67.             return total;
68.         }
69.         public void setTotal(long total) {
70.             this.total = total;
71.         }
```

```
72.     public List<? > getRows( ) {
73.         return rows;
74.     }
75.     public void setRows(List<? > rows) {
76.         this. rows = rows;
77.     }
78. }
```

代码说明：

- 第 3 行：调用 carMessageService 服务的 getAllCarMessage( ) 方法，将查询的结果存放于列表 carMessages 中，获取所有汽车的信息实现方法 getAllCarMessage 见步骤⑤。
- 第 4 行：创建 List 对象 carshows 用于存放查询的汽车信息。
- 第 5 行~第 16 行：如果获取的汽车列表不为空，则将每一条汽车信息获取出来，并添加到 carshow 列表中。
- 第 17 行：创建 TableReturnObject 对象，用于存放读取出来的汽车列表的相关信息，包括记录数、所有汽车信息。
- 第 18 行：将汽车记录数存放到 TableReturnObject 对象的 Total 字段中。
- 第 19 行：将获取的所有汽车信息存放到 TableReturnObject 对象的 rows 字段中。
- 第 20 行：将 tableReturnObject 中的数据转换成 JSON 格式，存放在 result 中。
- 第 21 行：请求获取参数"jsoncallback"。
- 第 22 行~第 23 行：设置网络文件的类型和网页的编码。
- 第 24 行~第 30 行：如果获取的结果不为空，则将结果返回。
- 第 31 行~第 62 行：定义 Carshow 实体类。
- 第 63 行~第 78 行：定义 TableReturnObject 实体类，用于定义要输出显示的汽车列表的相关信息。

⑤ 在业务逻辑的处理层类 CarMessageService. Java 文件中声明 getAllCarMessage 方法：

```
1.  public List<CarMessage> getAllCarMessage( ) {
2.          return carMessageMapper.getAllCarMessage( );
3.      }
```

⑥ 实现对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml。

打开 src 文件包 com. piesat. zyms. persistence 中 CarMessageMapper. java，添加定义 getAllCarMessage( ) 方法，获取所有的车辆数，代码如下。

```
1.  public List<CarMessage> getAllCarMessage( ) {
2.      return carMessageMapper. getAllCarMessage( );
3.  }
```

继续在刚才建的包文件 CarMessageMapper. xml 中添加映射数据表 CarMessage 的字段和表上的操作，代码如下。

```
1.  <select id="getAllCarMessage" resultType="com. piesat. zyms. domain. core. CarMessage">
2.      select * from carmessage
3.  </select>
```

任务 7.3　分页显示车辆信息

【任务描述】

当车辆的信息过多时，在同一页显示不便于查看，需要对车辆信息分页显示，用户可通过翻页来查看不同的汽车信息，效果如图 7-3-1 所示。

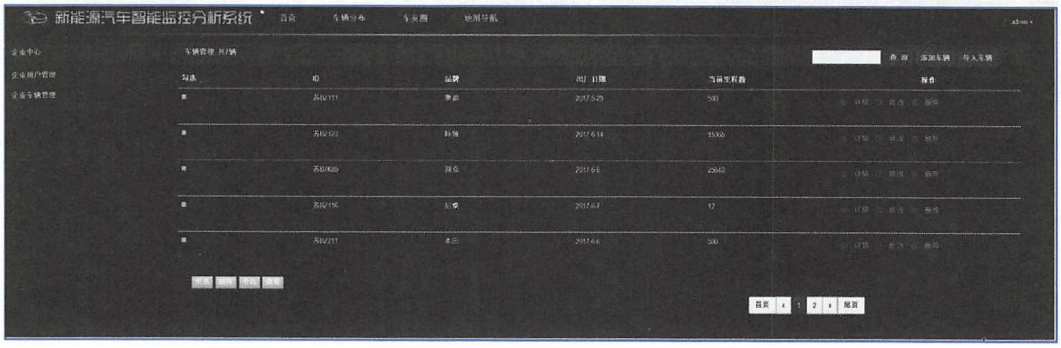


图 7-3-1　车辆信息分页显示

微课 7-5
车辆列表展示以及分页的实现

【任务目标】

知识目标

- 熟悉 MyBatis 框架架构和原理
- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法
- 熟悉 jqPaginator 分页插件的使用方法

技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据添加
- 使用 jqPageinator 实现车辆信息的分页显示

【任务分析】

当页面首次加载时，默认显示第 1 页汽车信息，同时在页面中显示总共的页数及当前的页码，用户可通过单击页码，跳转到相关页面的显示。具体的开发流程如下：

① 下载 jqPaginator 分页插件的 js 文件。

② 在页面中引入插件，并编写 Ajax 调用代码。

③ 实现 CarMessageController 中的分页检索，根据用户提交的要求将对应页显示。

【任务实施】

(1) 下载 jqPaginator. js 插件

在网络资源中 ajaxFileUpload. js 可以找到很多同名的插件，本任务插件的下载地址是：https://github. com/keenwon/jqPaginator/releases，下载页面如图 7-3-2 所示。

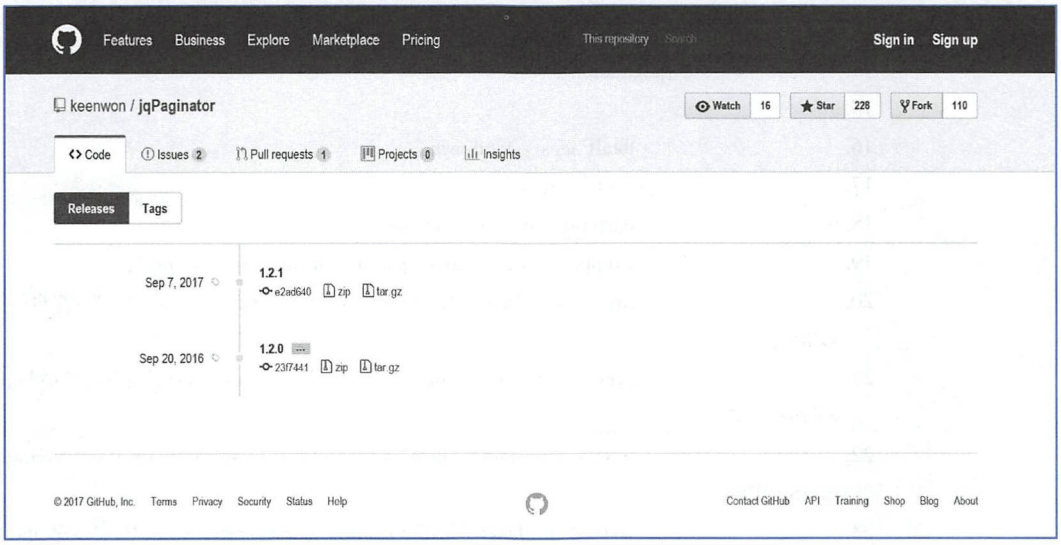


图 7-3-2　分页插件下载页面

(2) 在页面中引入插件，并编写 Ajax 调用代码

① 在 manage. jsp 页面中引入 jqPaginator 插件。引入插件的代码如下。

```
<script type="text/javascript" src="resources/js/jqPaginator.js"></script>
```

② 在 manage. jsp 页面中添加分页导航层。

```
<div class="col-md-6"  style="text-align: center;margin-top: 20px;">
    <ul id="pagination"></ul>
</div>
```

③ 在 manage. jsp 页面中编写 Ajax 分页代码。

创建页面加载的初始化函数 initpage()，设置分页的相关参数，同时在页面加载时进行方法的调用。实现代码如下：

```
1.  $(function(){
2.                  initpage();
3.          })
```

```
4.  function initpage() {
5.          var totalpage = "${pages}";
6.          totalpage = parseInt(totalpage);
7.          if (totalpage == 0)
8.          {
9.              return;
10.         }
11.         var currentpage = "1";
12.         currentpage = parseInt(currentpage);
13.         $.jqPaginator(
14.         '#pagination',
15.         {
16.             totalPages : totalpage,
17.             visiblePages : 5,
18.             currentPage : currentpage,
19.             wrapper : '<ul class="pagination lastspan"></ul>',
20.             first : '<li class="first"><a href="javascript:void(0);">首页</a>
    </li>',
21.             prev : '<li class="prev"><a href="javascript:void(0);">&laquo;
    </a></li>',
22.             next : '<li class="next"><a href="javascript:void(0);">&raquo;
    </a></li>',
23.             last : '<li class="last"><a href="javascript:void(0);">尾页</a>
    </li>',
24.             page : '<li class=""><a href="javascript:void(0);">{{page}}
    </a></li>',
25.             onPageChange : function(num){
26.                 Search(num);
27.             }
28.         });
29.     }
```

代码说明：

- 第 1 行~第 3 行：页面加载时，调用 initpage()。
- 第 4 行~第 29 行：定义 initpage()。
- 第 5 行：从控制层获取总页数 pages，此参数值在控制层进行获取。
- 第 6 行~第 10 行：获取总页数，如果页数为 0，则直接返回，否则获取总页数。
- 第 11 行~第 12 行：设置当前要显示的页。

- 第 14 行~第 24 行：初始化分页。对总页数 totalPages，最多显示的页码数 visiblePages（默认值为 7，例如有 100 页，当前第 1 页，则显示 1~7 页），当前的页码 currentPage，分页 HTML 结构属性 wrapper 及首页、上一页、下一页、末页的相关设置。
- 第 25 行~第 26 行：当换页时触发回调函数 search（num），此处需要将要显示的页码传递过去，需要修改任务 7.2 中的 search()函数。

修改任务 7.2 中的 search()函数，添加参数 num，根据用户选择显示对应的页面。修改后代码如下：

```
1.  function search(num){
2.              debugger;
3.              $.ajax({
4.              type:"post",
5.              url:'<%=path%>'+"/carmessage/search",
6.              dataType : "jsonp",
7.              jsonp:"jsoncallback",
8.              data:{
9.                  "pageNumber":num,
10.                 "pageSize":5,
11.                 },
12.             success : function(data){
13.                     ……
14.             });
15.         }
```

代码说明：

- 此处传递的参数为需要显示的页码 pageNumber 及每页显示的记录数 pagesize，此处设置为 5。

(3) 创建分页信息对应的 PageInfo 类 PageInfo.java

① 在 src 包中 com.piesat.zyms.utils 中新建一个类 PageInfo。

② 在类中创建属性：page（当前显示的页数）、pages（总页数）、size（总记录数）、len（每页显示的记录数）。

③ 创建好属性后，另起一行，右击，在弹出的快捷菜单中选择“source→Generate Getters and Setters”命令，对属性进行封装。

④ 创建方法 upPages()计算总页数。

⑤ 重写输出方法 toString()，右击，在弹出的快捷菜单中选择“source→Generate toString”命令，并添加代码。

实现代码如下：

```
1.  public class PageInfo {
2.      private long page=1;                    //当前页数 1 开始
```

```
3.      private long pages;                    //总页数
4.      private long size;                     //总的数据数
5.      private long len;                      //一页最多显示量
6.      public long getPage( ) {
7.          if( page<1) {
8.              page=1;
9.          }
10.         return page;
11.     }
12.     public void setPage(long page) {
13.         this. page = page;
14.     }
15.
16.     public long getPages( ) {
17.         return pages;
18.     }
19.     public long getSize( ) {
20.         return size;
21.     }
22.     public void setSize(long size) {
23.         this. size = size;
24.         upPages( );
25.     }
26.     public long getLen( ) {
27.         return len;
28.     }
29.     public void setLen(long len) {
30.         this. len = len;
31.         upPages( );
32.     }
33.     @ Override
34.     public String toString( ) {
35.         StringBuilder builder = new StringBuilder( );
36.         builder. append( " PageInfo [ page=" );
37.         builder. append( page );
38.         builder. append( " , pages=" );
39.         builder. append( pages );
40.         builder. append( " , size=" );
41.         builder. append( size );
42.         builder. append( " , len=" );
43.         builder. append( len );
44.         builder. append( " ]" );
45.         return builder. toString( );
```

```
46.     }
47.     private void upPages(){
48.         try {
49.             if(len<1){
50.                 len=1;
51.             }
52.             this.pages=size / len + (size % len > 0 ? 1 : 0);
53.         } catch (Exception e) {
54.         }
55.     }
56. }
```

代码说明：

- 第 2 行~第 32 行：设置属性并进行封装。
- 第 33 行~第 46 行：定义输出方法 toString ()。
- 第 47 行~第 55 行：定义 upPages()方法计算总页数，如果设置的每页显示记录数不足 1，则设置为 1，总页数为总记录数除以每页显示记录数，如余数不为零，则总页数加 1。

(4) 控制层用户分页的实现

① 在 CarMessageController 中修改 all，先计算出总分页数，并实现请求转发。

```
1. @RequestMapping("/all")
2.     public ModelAndView showCar(HttpServletRequest request,
3.             HttpServletResponse response, ModelMap model) {
4.         ModelAndView mv = new ModelAndView();
5.         mv.setViewName("car/manage.jsp");
6.         long count = carMessageService.getTotal();              //查询车辆总数
7.         List<User> users = userService.getUser();               //获取用户数
8.         long pages = count / 5 + (count % 5 > 0 ? 1 : 0);
9.         mv.addObject("pages", pages);
10.        mv.addObject("count", count);
11.        mv.addObject("users", users);
12.        return mv;
13.    }
```

代码说明：

- 调用 carMessageService 的 getTotal()方法查询车辆总数，并计算出总页数，本例中以每页显示 5 条记录为例，并将结果添加到 ModealMap 中去。

② 在 CarMessageController 中修改 search，接收参数 pageNumber、pageSize，实现请求转发。

```
14. @RequestMapping("/search")
15.     public void search(HttpServletRequest req, HttpServletResponse resp,
```

```
16.          @RequestParam("pageNumber") Integer pageNumber,
17.          @RequestParam("pageSize") Integer pageSize) throws IOException {
18.      PageInfo pageInfo = new PageInfo();
19.      pageInfo.setPage(pageNumber == null ? 1 : pageNumber);
20.      pageInfo.setLen(pageSize == null ? 1000 : pageSize);
21.      List<CarMessage> carMessages = carMessageService.searchCar(str,
22.              pageInfo);
23.      List<Carshow> carshows = new ArrayList<>();
24.      if (carMessages.size() > 0 && carMessages != null) {
25.          for (CarMessage carMessage : carMessages) {
26.              Carshow carshow = new Carshow();
27.              carshow.setVehID(carMessage.getVehID());
28.              carshow.setCurrentMileage(carMessage.getCurrentMileage());
29.              if (carMessage.getProduceDate() != null)
30.                  carshow.setProduceDate(carMessage.getProduceDate()
31.                          .getTime());
32.              carshow.setBrand(carMessage.getBrand());
33.              carshows.add(carshow);
34.          }
35.      }
36.      TableReturnObject tableReturnObject = new TableReturnObject();
37.      tableReturnObject.setTotal(carshows.size());
38.      tableReturnObject.setRows(carshows);
39.      String result = new Gson().toJson(tableReturnObject);
40.      String jsonp = req.getParameter("jsoncallback");
41.      resp.setCharacterEncoding("UTF-8");
42.      resp.setContentType("text/html");
43.      if (jsonp != null) {
44.          result = jsonp + "(" + result + ")";
45.          resp.getWriter().write(result);
46.      } else {
47.          resp.getWriter().write(result);
48.      }
49.  }
```

代码说明：

- 第 16 行～第 17 行：接收 View 传递的参数 pageNumber（当前页码）、pageSize（每页记录数）。
- 第 18 行～第 20 行：设置分页对象的属性 Page、Len。
- 第 21 行：按照用户请求的页码，调用 carMessageService 的 searchCar（）获取相应页码的记录数，具体在步骤（5）中实现。

(5) 实现业务逻辑的处理层类 carMessageService 的 searchCar( )

打开 src 文件夹下包 com. piesat. zyms. service. cms 下的类文件 carMessageService. java，添加方法 searchCar (PageInfo pageInfo)，代码如下：

```
1.  public List<CarMessage> searchCar( PageInfo pageInfo) {
2.          List<CarMessage> carMessages = new ArrayList<>( );
3.          if (pageInfo == null) {
4.              carMessages = carMessageMapper. getAllCarMessage( );
5.          } else {
6.              long count = carMessageMapper. queryNumber( );
7.              pageInfo. setSize( count);
8.              Map<String, Object> map=new HashMap<String, Object>( );
9.              map. put( "start", (pageInfo. getPage( ) - 1) * pageInfo. getLen( ));
10.             map. put( "end", pageInfo. getLen( ));
11.             carMessages = carMessageMapper. searchCarMessage( map);
12.         }
13.         return carMessages;
14.     }
```

代码说明：

- 第 2 行~第 5 行：如果接受的分页对象为空，则将所有汽车信息读取到 carMessages 中。
- 第 6 行~第 12 行：如果接受的分页对象不为空，则按要求读取所要检索页的记录。
- 第 6 行：通过 queryNumber( )获取汽车记录总数，存放在 count 中。
- 第 8 行~第 10 行：将参数起始记录 start，结束记录号 end，压入 Map 对象中。
- 第 11 行：通过 searchCarMessage( )获取所要的记录。

(6) 实现对象持久化映射层 carMessageMapper. java、carMessageMapper. xml

① 打开 src->com. piesat. zyms. persistence->carMessageMapper. java。

② 定义查询汽车记录数方法 queryNumber、按分页信息查询汽车 searchCarMessage，代码如下：

```
package com. piesat. zyms. persistence;
public interface CarMessageMapper {
    public long queryNumber( );
    public List<CarMessage> searchCarMessage( Map<String, Object> map);
}
```

③ 在 carMessageMapper. xml 文件中，增加 queryNumber、searchCarMessage 定义，代码如下：

```
1.  <select id="queryNumber" resultType="java. lang. Long">
2.          select count( * ) from carmessage
```

```
3.    </select>
4.    <select id="searchCarMessage"  parameterType="java.lang.Long"
5.                    resultType="com.piesat.zyms.domain.core.CarMessage">
6.            select  *  from      carmessage      order by id limit #{start},#{end}
7.    </select>
```

代码说明：

- 第 1 行~第 3 行：查询汽车记录数。
- 第 4 行~第 7 行：获取指定范围的记录。

任务 7.4　显示车辆详情

【任务描述】

在车辆信息显示页面（manage.jsp）中，通过单击“详情”按钮可进入汽车详细情况页面（CarInformation.jsp），查看汽车故障信息，效果如图 7-4-1 和图 7-4-2 所示。

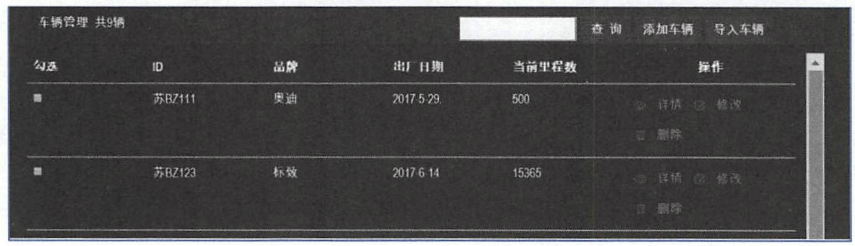


图 7-4-1　车辆信息显示页面

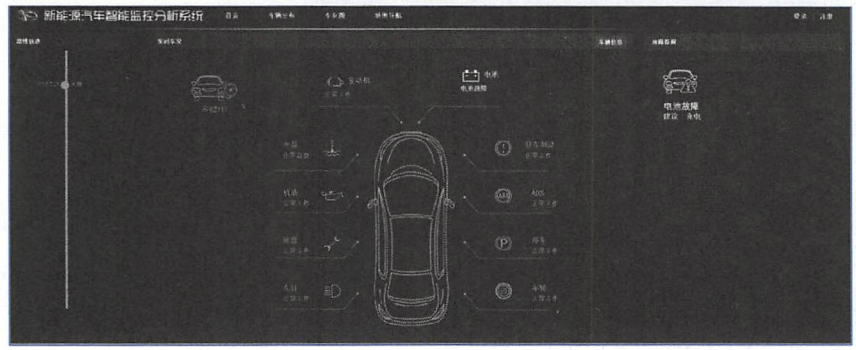


图 7-4-2　车辆详情页面

【任务目标】

知识目标

- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法。
- 熟悉 EL 和 JSTL 的使用方法。

技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据显示。
- 使用 EL 和 JSTL 显示数据。

【任务分析】

微课 7-6
新增车辆和查询车辆详情

用户单击“详情”按钮后，进入汽车详情页面，如图 7-4-2 所示。页面主要分成 3 个显示部分，左侧显示车辆行程轨迹，中部显示车辆实时故障信息，右侧显示故障提醒。这 3 个部分的数据分别存放在数据表 tbvehlocation、faultinfo、tbfaultmean、tbfaultsolution 中。使用框架技术在各层中具体实现思路如下。

- 创建各表所对应的实体类：车辆信息类 FaultInfo、行程轨迹类 CarLocation、故障说明类 Faultmean、故障建议类 FaultSolution。
- 在 manage. jsp 页面单击“详情”按钮触发 View( )方法，转向 Controller 层。
- 编辑控制层 IndexController. java，添加 carinformation( )方法。
- 完成业务逻辑的处理层类 FaultInfoService. java、CarLocationService. Java、FaultSolutionService。
- 完成对象持久化映射层类 FaultInfoMapper. java、CarLocationMapper. java、FaultSolutionMapper.java、FaultInfoMapper. xml、CarLocationMapper. xml、FaultSolutionMapper. xml。

【任务实施】

(1) 创建车辆信息类、行程轨迹类、故障提醒类、故障说明等实体类

① 在文件夹 src 的 com. piesat. zyms. domain. core 包中创建故障说明类 Faultmean，在数据库中故障相关的说明信息存放在 tbfaultmean 表中，类的属性与表的字段一一对应，如图 7-4-3 所示。

名	类型	长度
Id	int	11
faultID	varchar	50
faultmean	varchar	50
remarks	varchar	255

(a)

```
public class Faultmean {
    private String Id;

    private String faultID;

    private String faultmean;

    private String remarks;
```

(b)

图 7-4-3　tbfaultmean 表字段与 Faultmean 类属性对应图

代码如下：

```
1. public class Faultmean {
2.     private String Id;
3.     private String faultID;
4.     private String faultmean;
5.     private String remarks;
6.     public String getFaultID( ) {
7.         return faultID;
8.     }
9.     public void setFaultID(String faultID) {
```

```
10.             this. faultID = faultID;
11.         }
12.         public String getFaultmean( ) {
13.             return faultmean;
14.         }
15.         public void setFaultmean( String faultmean) {
16.             this. faultmean = faultmean;
17.         }
18.         public String getRemarks( ) {
19.             return remarks;
20.         }
21.         public void setRemarks( String remarks) {
22.             this. remarks = remarks;
23.         }
24.         public String getId( ) {
25.             return Id;
26.         }
27.         public void setId( String id) {
28.             Id = id;
29.         }
30. }
```

② 在文件夹 src 的 com. piesat. zyms. domain. core 包中创建故障提醒类 FaultSolution，用于存放故障对应的提醒信息，在数据库中故障对应提醒信息存放在 tbfaultsolution 表中，类的属性与表的字段一一对应，如图 7-4-4 所示。

名	类型	长度
id	int	11
faultID	varchar	100
verNumber	int	2
faultSolution	varchar	500
remarks	varchar	255

(a)

```
public class FaultSolution {
    private String id;
    private String faultID;
    private String verNumber;
    private String faultSolution;
    private String remarks;
```

(b)

图 7-4-4　tbfaultsolution 表字段与 FaultSolution 类属性对应图

代码如下：

```
1. public class FaultSolution {
2.     private String id;
3.     private String faultID;
4.     private String verNumber;
5.     private String faultSolution;
6.     private String remarks;
7.     public String getId( ) {
8.         return id;     }
```

```
9.      public void setId(String id) {
10.         this.id = id;
11.     }
12. public String getFaultID() {
13.         return faultID;
14.     }
15.     public void setFaultID(String faultID) {
16.         this.faultID = faultID;
17.     }
18.     public String getVerNumber() {
19.         return verNumber;
20.     }
21.     public void setVerNumber(String verNumber) {
22.         this.verNumber = verNumber;
23.     }
24.     public String getFaultSolution() {
25.         return faultSolution;
26.     }
27.     public void setFaultSolution(String faultSolution) {
28.         this.faultSolution = faultSolution;
29.     }
30.     public String getRemarks() {
31.         return remarks;
32.     }
33.     public void setRemarks(String remarks) {
34.         this.remarks = remarks;
35.     }
36. }
```

③ 在文件夹 src 的 com.piesat.zyms.domain.core 包中创建故障说明类 CarLocation，在数据库中故障相关的说明信息存放在 tbvehlocation 表中，类的属性与表的字段一一对应，如图 7-4-5 所示。

名	类型	长度
id	int	11
vehID	varchar	100
modID	varchar	100
realtime	varchar	100
latitudes	varchar	255
longitudes	varchar	255
remarks	varchar	255
recordtime	date	0
province	varchar	255

(a)

```
public class CarLocation {
    private Integer id;
    private String vehID;
    private String modID;
    private String realtime;
    private String latitudes;
    private String longitudes;
    private String remarks;
    private Date remarktime;
    private int count;
    private String province;
    private String plateNumber;
```

(b)

图 7-4-5　tbvehlocation 表字段与 CarLocation 类属性对应图

④ 在文件夹 src 的 com. piesat. zyms. domain. core 包中创建故障信息显示类 FaultInfo，在数据库中故障相关的说明信息存放在 faultinfo 表中，本任务是根据车辆的牌号显示该车相关的故障名称、故障提醒等相关信息，除了类的属性与表的字段一一对应外，另添加属性 Faultmean、FaultSolution，如图 7-4-6 所示。

名	类型	长度
id	int	11
faultID	varchar	50
faultState	varchar	100
vehID	varchar	50
realtime	varchar	100
faultLevel	varchar	50
modID	varchar	50
remarks	varchar	255

(a)

```
public class FaultInfo {
    private Integer id;

    private String faultID;
    private String faultState;
    private String faultLevel;
    private String vehID;
    private String modID;
    private String realtime;
    private String remarks;
    private Faultmean faultmean;
    private FaultSolution faultsolution;

    private int fid;
    private int count;
    private String fmodID;
    private String fremarks;
```

(b)

图 7-4-6　faultinfo 表字段与 FaultInfo 类属性对应图

(2) 在 manage. jsp 页面实现“详情”调用

① 在任务 7.2 的车辆信息显示中，已经为“详情”绑定了 onclick 事件 View (vehID)，单击“详情”将触发相关事件。

```
html+="<td style='width: 22%;'><div style='float: left;'><img alt='del' src='resources/images/view.png'><a class=\"btn btn-link editdeptbtn\" style='color: #20c8ff' onclick=\"view('"+listdata[f].vehID+"');\">详情</a></div>";
```

② 在 manage. jsp 页面中添加 view () 方法的定义，传递参数 vehID 车辆编号，将请求发送到 Control 层，代码如下。

```
1. function view(vehID){
2.     var id = encodeURI(encodeURI(vehID));
3.     window.location.href='http://' + location.host + '<%=path%>/'+ 'energy/carinformation/' + id;
4. }
```

代码说明：

- 第 1 行：参数 vehID 代表车辆编号。
- 第 2 行：使用 encodeURI() 来编码车辆编号。
- 第 3 行：请求发送到控制层，同时传递参数车辆编号。

(3) IndexController 中实现"/energy/carinformation"

在 IndexController 中新建方法，具体实现思路为：根据用户选择获取车牌号，并通过 Service 层分别获取该车辆对应的行程轨迹信息、故障信息、故障提醒信息，并将结果回送到 View 层，代码如下。

```
1. @RequestMapping(value = "/carinformation/{vehID}", method = RequestMethod.GET)
2.     public ModelAndView carinformation(HttpServletRequest request,
3.             HttpServletResponse response, ModelMap model,@PathVariable String ve-
   hID) throws UnsupportedEncodingException {
4.         String vid = URLDecoder.decode(vehID,"UTF-8");
5.         String fdj = "0";                    //发动机
6.         String dc = "0";                     //电池
7.         String sw = "0";                     //水温
8.         String zczd = "0";                   //驻车制动
9.         String jy = "0";                     //机油
10.        String dp = "0";                     //底盘
11.        String abs = "0";                    //abs
12.        String park = "0";                   //停车
13.        String light = "0";                  //车灯
14.        String wheel = "0";                  //车轮
15.        CarMessage cm = carMessageService.getCarMessageByVehID(vid);
16.        List<CarLocation> cl = carLocationService.getCarMessageByVehID(vid);
17.        List<FaultInfo> fi = faultInfoService.getfaultinfoByVehID(vid);
18.        ArrayList<FaultSolution> fs = new ArrayList<>();
19.        for (FaultInfo fl : fi) {
20.            if (fl.getFaultState().equals("发动机")) {
21.                fdj = fl.getFaultID();
22.            }
23.        if (fl.getFaultState().equals("电池")) {
24.                dc = fl.getFaultID();
25.            }
26.            if (fl.getFaultState().equals("水温")) {
27.                sw = fl.getFaultID();
28.            }
29.            if (fl.getFaultState().equals("驻车制动")) {
30.                zczd = fl.getFaultID();
31.            }
32.            if (fl.getFaultState().equals("机油")) {
33.                jy = fl.getFaultID();
34.            }
35.            if (fl.getFaultState().equals("底盘")) {
36.                dp = fl.getFaultID();
37.            }
38.            if (fl.getFaultState().equals("abs")) {
39.                abs = fl.getFaultID();
40.            }
41.            if (fl.getFaultState().equals("停车")) {
42.                park = fl.getFaultID();
```


```
43.                 }
44.                 if (fl.getFaultState().equals("车轮")) {
45.                     wheel = fl.getFaultID();
46.                 }
47.             FaultSolution fsl = faultSolutionService.getSolutionByFaultID(fl.getFaultID
                ());
48.                 fs.add(fsl);
49.             }
50.             ModelAndView mv = new ModelAndView();
51.             mv.setViewName("car/CarInformation.jsp");
52.             mv.addObject("carmessage", cm);
53.             if (cl != null) {
54.                 mv.addObject("carlocation", cl);
55.             }
56.             if (fi != null) {
57.                 mv.addObject("fdj", fdj);
58.                 mv.addObject("dc", dc);
59.                 mv.addObject("sw", sw);
60.                 mv.addObject("zczd", zczd);
61.                 mv.addObject("jy", jy);
62.                 mv.addObject("abs", abs);
63.                 mv.addObject("dp", dp);
64.                 mv.addObject("park", park);
65.                 mv.addObject("light", light);
66.                 mv.addObject("wheel", wheel);
67.             }
68.             mv.addObject("faultsolution", fs);
69.             return mv;
70.     }
```

代码说明：

- 第 4 行：获取车辆车牌信息，并将 View 层传递的车牌号进行解码。
- 第 5 行~第 14 行：初始化车辆各部件（发动机、电池、水温、驻车制动、机油、底盘、ABS、停车、车灯、车轮）的状态值，无异常则为 0。
- 第 15 行：调用 carMessageService 服务的 getCarMessageByVehID（id）方法根据车牌号获取指定车辆信息，存放在 cm 对象中，方法的具体实现在步骤（4）中介绍。
- 第 16 行：调用 carLocationService 服务的 getCarMessageByVehID（vid）方法根据车牌号获取车辆行程轨迹信息，存放在 c1 对象中。
- 第 17 行：调用 faultInfoService 服务的 getfaultinfoByVehID（vid）方法根据车牌号获取车辆故障信息，存放在 fi 对象中。
- 第 18 行：创建故障提醒数据列表对象 fs，用于存放车辆故障信息集合。
- 第 19 行~第 49 行：从获取的故障信息列表中，逐个获取车辆各个零件的

故障状态，如果获取到该零件的故障码，则修改该零件对应的故障状态值。

● 第 50 行：将查得指定车辆的故障诊断提醒记录读取出来存放在 FaultSolution 类的对象 fsl 中。

第 51 行~第 69 行：使用 ModelAndView 进行重定向到"car/CarInformation.jsp"，并将各部件（发动机、电池、水温、驻车制动、机油、底盘、ABS、停车、车灯、车轮）的状态值、车辆基本信息、车辆行程轨迹、车辆故障提醒等相关数据传递到 View 层页面中。

（4）车辆故障显示详情页数据显示的业务逻辑的处理层及对象持久化映射层的实现

根据车辆车牌分别获取车辆行程轨迹相关信息、车辆故障相关信息、车辆基础信息及车辆故障提醒等功能的实现。

① 显示车辆基础信息的业务逻辑层（carMessageService.java）及对象持久化映射层（carMessageMapper.java、carMessageMapper.xml）的实现。

在 carMessageService.java 中添加 getCarMessageByVehID（vid）方法，通过该方法实现根据车牌号获取汽车的信息，代码如下。

```
1. public CarMessage getCarMessageByVehID(String vehID) {
2.         return carMessageMapper.getCarMessageByVehID(vehID);
3.     }
```

CarMessageMapper.java 文件中代码如下：

```
1. public interface CarMessageMapper {
2. ……
3.     /**
4.      * 根据 vehID 获取车辆
5.      */
6.     public CarMessage getCarMessageByVehID(String vehID);
7. }
```

在 CarMessageMapper.xml 文件中添加如下代码。

```
<select id="getCarMessageByVehID" parameterType="java.lang.String" resultType="com.piesat.zyms.domain.core.CarMessage">
        select * from carmessage where vehID like #{vehID}
    </select>
```

② 车辆行程轨迹的显示业务逻辑层（carLocationService.java）及对象持久化映射层（carLocationMapper.java、carLocationMapper.xml）的实现。

● 在 src 文件夹的 com.piesat.zyms.service.cms 包下新建类 carLocationService.java，在类中添加方法 getCarMessageByVehID（vid），通过该方法实现获取行程轨迹的相关信息，代码如下。

```
1. @Service
2. public class CarLocationService {
3.     @Autowired
```


```
4.      private CarLocationMapper carLocationMapper;
5.      public List<CarLocation> getCarMessageByVehID(String vehID) {
6.          return carLocationMapper.getCarMessageByVehID(vehID);
7.      }
8. }
```

● 在 src 文件夹的 com.piesat.zyms.persistence 包下新建类 CarLocationMapper.java，并在文件中添加 getCarMessageByVehID（vehID）方法；利用在 carLocationMapper.xml 中配置的 SQL 语句进行数据的读取。CarLocationMapper .java 文件中的代码如下。

```
1. public interface CarLocationMapper {
2.     /**
3.      * 根据 vehID 获取车辆
4.      * @param userid
5.      * @return
6.      */
7.     public List<CarLocation> getCarMessageByVehID(String vehID);
8. }
```

CarLocationMapper.xml 文件中的代码如下。

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
3. <mapper namespace="com.piesat.zyms.persistence.CarLocationMapper">
4.     <resultMap id="BaseResultMap" type="com.piesat.zyms.domain.core.CarLocation">
5.         <id column="id" property="id" />
6.         <result column="vehID" property="vehID" />
7.         <result column="modID" property="modID" />
8.         <result column="realtime" property="realtime" />
9.         <result column="latitudes" property="latitudes" />
10.        <result column="longitudes" property="longitudes" />
11.        <result column="remarks" property="remarks" />
12.        <result column="remarktime" property="remarktime" />
13.        <result column="count" property="count" />
14.        <result column="provid" property="provid" />
15.        <result column="plateNumber" property="plateNumber" />
16.    </resultMap>
17.    <select id="getCarMessageByVehID" parameterType="java.lang.String" resultType="com.piesat.zyms.domain.core.CarLocation">
18.        select * from tbvehlocation where vehID = #{vehID}
19.    </select>
20. </mapper>
```

③ 车辆故障相关信息显示业务逻辑层（FaultInfoService. java）及对象持久化映射层（FaultInfoMapper. java、FaultInfoMapper. xml）的实现。

● 在 src 文件夹的 com. piesat. zyms. service. cms 包下新建类 FaultInfoService. java，在类中添加 getfaultinfoByVehID（vehID）方法，通过该方法实现根据车牌号获取车辆相关故障记录，代码如下。

```
1. @Service
2. public class FaultInfoService {
3.     @Autowired
4.     private FaultInfoMapper faultInfoMapper;
5.     public List<FaultInfo> getfaultinfoByVehID(String vehID) {
6.         return faultInfoMapper.getfaultinfoByVehID(vehID);
7.     }
```

● 在 src 文件夹的 com. piesat. zyms. persistence 包下新建类 FaultInfoMapper. java，并在文件中添加 getfaultinfoByVehID（vehID）方法，实现根据车牌号检索车辆故障信息的方法；利用在 faultInfoMapper. xml 中配置的 SQL 语句进行车辆故障数据的读取。FaultInfoMapper. java 文件中的代码如下。

```
1. public interface FaultInfoMapper {
2.     /** * 根据 vehID 获取车辆 */
3.     public List<FaultInfo> getfaultinfoByVehID(String vehID);
4. }
```

FaultInfoMapper . xml 文件中的代码如下：

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://myb-
   atis.org/dtd/mybatis-3-mapper.dtd">
3. <mapper namespace="com.piesat.zyms.persistence.FaultInfoMapper">
4.     <resultMap id="BaseResultMap" type="com.piesat.zyms.domain.core.FaultInfo">
5.         <id column="id" property="id" />
6.         <result column="vehID" property="vehID" />
7.         <result column="faultID" property="faultID" />
8.         <result column="faultState" property="faultState" />
9.         <result column="faultLevel" property="faultLevel" />
10.        <result column="modID" property="modID" />
11.        <result column="realtime" property="realtime" />
12.        <result column="remarks" property="remarks" />
13.        <result column="count" property="count" />
14.        <result column="fremarks" property="fremarks" />
15.        <result column="fmodID" property="fmodID" />
16.        <result column="fid" property="fid" />
```

```
17. <select id="getfaultinfoByVehID" parameterType="java.lang.String"
18. resultType="com.piesat.zyms.domain.core.FaultInfo">
        select * from faultinfo where vehID = #{vehID}
19.     </select>
```

④ 车辆故障提醒显示业务逻辑层（FaultInfoService.java）及对象持久化映射层（FaultInfoMapper.java、FaultInfoMapper.xml）的实现。

● 在 src 文件夹的 com.piesat.zyms.service.cms 包下新建类 FaultSolutionService.java，在类中添加 getSolutionByFaultID（faultID）方法，通过该方法实现根据故障码来获取车辆相关故障提醒信息，代码如下。

```
1. @Service
2. public class FaultSolutionService {
3.     @Autowired
4.     private FaultSolutionMapper faultSolutionMapper;
5.     public FaultSolution getSolutionByFaultID(String faultID){
6.         return faultSolutionMapper.getSolutionByFaultID(faultID);
7.     }
8. }
```

● 在 src 文件夹的 com.piesat.zyms.persistence 包下新建类 FaultSolutionMapper.java，并在文件中添加 getSolutionByFaultID（faultID）方法，实现根据故障号检索车辆故障提醒信息；利用在 FaultSolutionMapper.xml 中配置的 SQL 语句读取车辆故障提醒。FaultSolutionMapper.java 文件中的代码如下。

```
public interface FaultSolutionMapper {
    public FaultSolution getSolutionByFaultID(String faultID);
}
```

FaultSolutionMapper.xml 文件中的代码如下：

```
<? xml version="1.0" encoding="UTF-8"? >
<! DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.piesat.zyms.persistence.FaultSolutionMapper">
    <resultMap id="BaseResultMap" type="com.piesat.zyms.domain.core.FaultSolution">
        <id column="id" property="id" />
        <result column="faultID" property="faultID" />
        <result column="verNumber" property="verNumber" />
        <result column="faultSolution" property="faultSolution" />
        <result column="remarks" property="remarks" />
</resultMap>
    <select id="getSolutionByFaultID" resultType="com.piesat.zyms.domain.core.FaultSolution">
        select * from tbfaultsolution where faultID = #{faultID}
```

```
    </select>
</mapper>
```

（5）通过 EL 表达式及 JSTL 标签库显示在 CarInformation. jsp 页面中车辆行程信息、车辆故障信息、车辆基本信息、车辆故障提醒等信息。

① 在 CarInformation. jsp 页面中引入 JSTL core 标签库。

- 下载 JSTL 包，并将下载的 standard-1. 1. 2. jar 和 jstl-1. 2. jar 包保存至项目的 lib 目录下，如图 7-4-7 和图 7-4-8 所示。

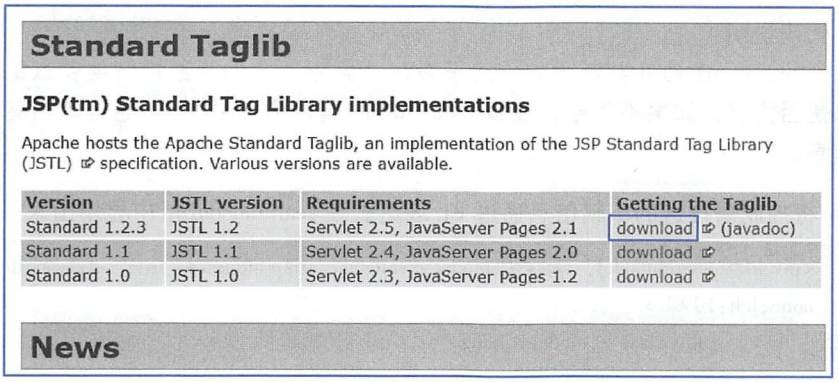


图 7-4-7　JSTL 包下载

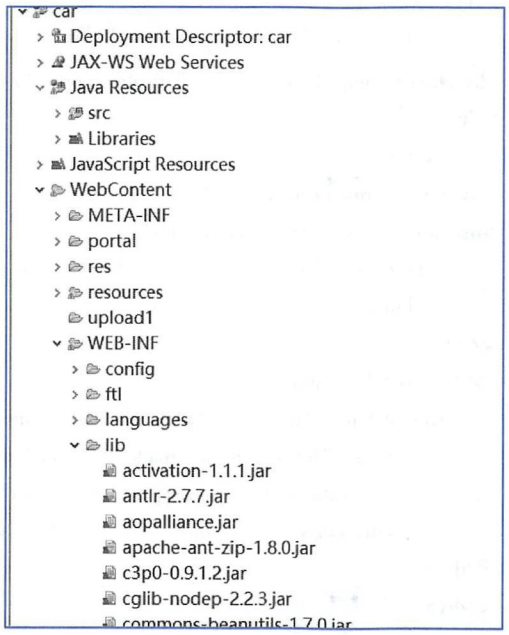


图 7-4-8　将包保存至项目的 lib 中

- 在页面中引入核心包。

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
```

② 显示车辆信息。

● 为“车辆信息”添加超链接，当单击“车辆信息”按钮时可弹出车辆信息层 searchmodal。

```
1. <c:if test="${not empty carmessage}">
2. <a class="btn btn-primary btn-sm" data-toggle="modal"  style="float: right;margin-top: -5px;" data-target="#searchmodal">车辆信息</a>
3. </c:if>
```

代码说明：

● <c: if >类似 if 判断语句，用于表达式判断。此处判断控制层获取的车辆信息是否为空，如果不为空，设置 data-toggle 属性为 modal，可通过单击弹出车辆信息。

● 将控制层推送的车辆信息通过 EL 表达式在 searchmodal 层中显示出来。

```
1. <div id="searchmodal" class="modal mymodal" aria-hidden="true" style="display: none;left:10%">
2.         <div class="modal-dialog">
3.             ......
4.             <div class="col-md-7 controls">
5.             <input type="text" id="vehID" name="vehID" class="form-control"
6.                     readonly="readonly" value="${carmessage.id}">
7.             <div class="help-block" style="display:none;"></div>
8.             </div>
9.                 </div>
10.            <div class="row form-group">
11.            <div class="col-md-3 control-label">
12.                 <label for="plateNumber">车牌号<span class="xb"></span>
                    </label>
13.            </div>
14.            <div class="col-md-7 controls">
15.                 <input type="text" id="plateNumber" name="plateNumber"
16.                     class="form-control" readonly="readonly"
17.                         value="${carmessage.plateNumber}">
18.                     <div class="help-block" style="display:none;"></div>
19.            </div>
20.            </div>
21.        <div class="row form-group">
22.            <div class="col-md-3 control-label">
23.                 <label for="brand">品牌<span class="xb"></span></label>
24.            </div>
25.            <div class="col-md-7 controls">
```

```
26.                 <input type="text" id="brand" name="brand" class="form-con-
                    trol"
27.                         readonly="readonly" value="${carmessage.brand}">
28.             <div class="help-block" style="display:none;"></div>
29.         </div>
30.         </div>
31.         <div class="row form-group">
32.             <div class="col-md-3 control-label">
33.             <label for="currentMileage">当前里程数</label>
34.             </div>
35.         <div class="col-md-7 controls">
36.             <input type="text" id="currentMileage" name="currentMileage"
37.                 class="form-control" readonly="readonly"
38.                 value="${carmessage.currentMileage}">
39.             <div class="help-block" style="display:none;"></div>
40.         </div>
41.         </div>
42.         <div class="row form-group">
43.             <label for="produceDate" class="col-md-3 control-label">出厂日期
                </label>
44.             <div class="col-md-7 controls"
45.                 style="margin-top: 5px;margin-left: 10px;">
46.                 <fmt:setLocale value="zh" />
47.                 <fmt:formatDate value="${carmessage.produceDate}" />
48.             </div>
49.         </div>
50.         <div class="row form-group">
51.             <div class="col-md-3 control-label">
52.                 <label for="userId">归属人</label>
53.             </div>
54.             <input id="username" class="form-control"
55.                 value="${user.username}" style="margin-left: 16px;"
56.             readonly="readonly"><input id="userid"
57.             value="${user.id}" type="hidden">
58.         </div>
59.         <div class="row form-group">
60.             <div class="col-md-3 control-label">
61.             <label for="remarks">说明</label>
62.             </div>
63.             <div class="col-md-7 controls">
```

```
64.                 <input type="text" id="remarks" name="remarks"
65.                         class="form-control" readonly="readonly"
66.                         value="${carmessage.distribution}">
67.                 <div class="help-block" style="display:none;"></div>
68.                 </div>
69.             </div>
70.             </form>
71.             </div>
72.             ......
73.         </div>
```

代码说明：

- EL 用于查找作用域中的数据，对其进行简单操作，本页面中通过 EL 表达式(${})来获取 CarMessage 实体类属性，${carmessage.id}即表示获取对象 carmessage 的 id 的属性值，其他字段显示以此类推。

③ 显示车辆行程信息。

从控制层中读取到的数据汽车行程轨迹存放在 carlocation 对象中，将数据一条一条读取出来，需要使用 JSTL 的 forEach 来实现循环遍历，结构如下。

<c：forEach var="对象" items="保存在 request 中的数组" varStatus="status">循环的变量<c：forEach />中。

具体实现代码如下。

```
1.  <div class="bheight" style="background-color: #145885;">
2.          <div id="carlist" style="height:500px;">
3.              <img alt="line" src="resources/images/myself/line2.png"
4.                                      style="margin-left: 50%;z-index: -1">
5.      <c:if test="${not empty carmessage}">
6.          <div>
7.              <ul class="guijipos">
8.          <c:forEach items="${carlocation}" var="carlocation" varStatus="cl">
9.              <div style="margin-left: -37px;margin-top: 30px">
10.             <li><span style="color:#20c8ff ">${carlocation.realtime}</span>
11.             <img alt="point" src="resources/images/myself/point.png">
12.             <span style="color: #20c8ff">${carlocation.remarks}</span></li>
13.             </div>
14.         </c:forEach>
15.             </ul></div>
16.         </c:if>
17.             </div>
18.         </div>
```

代码说明：

- 第 5 行：判断从控制层返回的 carmessage 中是否有信息。
- 第 8 行~第 14 行：获取该车辆的行程轨迹数据，将行程的时间、地点显示在指定轴上。

④ 获取车辆的故障各部件（发动机、电池、水温、驻车制动、机油、底盘、ABS、停车、车灯、车轮）的状态值，如果为 0，则正常显示，否则显示为不同的颜色，并给出提示信息，在此需要对状态进行分支处理，使用 JSTL 的<c:if>。

```
1.  <div class="col-lg-9 carimg">
2.          <c:if test="${fdj != '0'}">
3.              <div class="fdj">
4.                  <img alt="fdj2" src="resources/images/myself/fdj2.png">
5.                  <span style="color: #ffee62;font-size: 20px">发动机</span><br />
6.                  <span style="color: #ffee62;font-size: 18px">故障:请及时检修</
                    span>
7.              </div>
8.          </c:if>
9.          <c:if test="${fdj == '0'}">
10.             <div class="fdj">
11.             <img alt="fdj2" src="resources/images/myself/fdj1.png">
12.             <span style="color: #20c8ff;font-size: 20px">发动机</span><br />
13.             <span style="color: #20c8ff;font-size: 18px">正常工作</span>
14.             </div>
15.         </c:if>
16. <c:if test="${dc != '0'}">
17.             <div class="dc">
18. <img alt="fdj2" src="resources/images/myself/dc2.png">
19. <span style="color : #ffee62;font-size: 20px">电池 </span><br />
20. <span style="color: #ffee62;font-size: 18px">电池 故障</span>
21.         </div>
22. </c:if>
23. <c:if test="${dc == '0'}">
24.         <div class="dc">
25.         <img alt="fdj2" src="resources/images/myself/dc1.png">
26. <span style="color: #20c8ff;font-size: 20px">电池</span><br />
27. <span style="color: #20c8ff;font-size: 18px">正常工作</span>
28.         </div>
29.     </c:if>
30. <c:if test="${sw == '0'}">
```

```
31.              <div class="sw">
32.          .....
33.          </div>
34.          </c:if>
35.  <c:if test="${sw ! = '0'}">
36.              .....
37.          </c:if>
38.  <c:if test="${zczd == '0'}">
39.                  .....
40.
41.          </c:if>
42.  <c:if test="${zczd ! = '0'}">
43.                  .....
44.  </c:if>
45.  <c:if test="${jy == '0'}">
46.              .....
47.  </c:if>
48.  <c:if test="${jy ! = '0'}">
49.              .....
50.          </c:if>
51.          <c:if test="${abs == '0'}">
52.                          .....
53.  </c:if>
54.  <c:if test="${abs ! = '0'}">
55.           .....
56.  </c:if>
57.  <c:if test="${dp == '0'}">
58.                          .....
59.          </c:if>
60.  <c:if test="${dp ! = '0'}">
61.                  .....
62.  </c:if>
63.  <c:if test="${park == '0'}">
64.          .....
65.  </c:if>
66.  <c:if test="${park ! = '0'}">
67.                  .....
68.  </c:if>
69.  <c:if test="${light == '0'}">
70.                  .....
71.  </c:if>
72.  <c:if test="${light ! = '0'}">
73.                  .....
```

```
74. </c:if>
75. <c:if test="${wheel == '0'}">
76.                         .....
77. </c:if>
78. <c:if test="${wheel ! = '0'}">
79.                     .....
80. </c:if>
81.             </div>
82.             </div>
83.         </c:if>
84. </div>
85.         </div>
```

代码说明：

- 第 2 行~第 8 行：如果发动机的指示状态不为 0，说明发动机故障，则显示黄色的提示文字。
- 第 9 行~第 15 行：如果发动机的指示状态为 0，说明发动机正常，则显示蓝色的文字。
- 第 16 行~第 85 行：其他零部件的故障状态同样如此判断及显示，由于篇幅有限，略去各层设置代码。

⑤ 故障提醒显示。

从控制层获取的车辆故障提醒信息存放在 faultsolution 对象中，通过<c：forEach>遍历读取，代码如下。

```
1. <div class="bheight" style="background-color: #145885;">
2.             <c:if test="${not empty carmessage }">
3.         <img alt="guzhang" src="resources/images/myself/fault.png" style="margin-top:
   50px;margin-left: 50px">
4.         <div id="carlist" style="height:500px;">
5. <c:forEach items="${faultsolution}" var="fs" varStatus="fst">
6.          <div style="color: #ffee62;margin-left: 50px;margin-top: 20px;"><h3>${
   fs.remarks}</h3>
7.             <span style="font-size: 20px">建议:${fs.faultSolution}</span>
8.             </div>
9.         </c:forEach>
10.             </div>
11. </c:if>
12.         </div></div></div></div>
13.         </div>
```

代码说明：

- 第 2 行：判断车辆信息是否为空。
- 第 5 行~第 9 行：遍历 faultsolution，循环读出车辆故障提醒的信息。

任务 7.5　更新车辆信息

【任务描述】

通过 Spring+SpringMVC+MyBatis 框架实现车辆信息修改页面，效果如图 7-5-1 和图 7-5-2 所示。

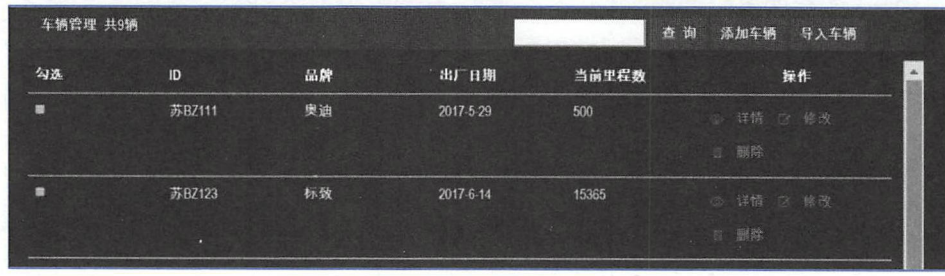


图 7-5-1　“修改”按钮

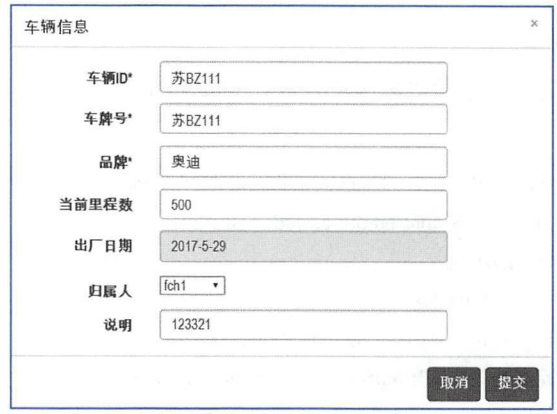


图 7-5-2　修改车辆信息

微课 7-7
车辆信息的更新和删除

【任务目标】

知识目标

- 熟悉 MyBatis 框架架构和原理。
- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法。

技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据修改。

【任务分析】

修改车辆信息的流程：单击“修改”按钮，弹出修改车辆页面，页面中显示当前选择的车辆的信息，修改信息后，单击“确定”按钮，可将修改的信息更新到数据表中。具体开发流程如下。

① 为“修改”按钮添加触发事件，在弹出层中显示要修改的汽车信息。

- 在 manage. jsp 页面为“修改”按钮添加触发事件，将请求发送到控制层。
- 在 CarMessageController 中创建汽车信息类 carDetailBean，便于控制层到 View 层数据传递。
- 在 CarMessageController 中创建 carDetail（vehID）方法，可根据指定车辆的车牌号获取车辆的详细信息。
- 在业务逻辑的处理层类 CarMessageService. Java，声明 carDetail 方法。
- 对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 detail（vehId）。

② 将修改后的汽车信息更新到数据库。

- 为“提交”按钮添加触发事件 changeCar()，将用户数据提交到控制层。
- 在 CarMessageController 中创建修改信息方法 updateCar()。
- 创建业务逻辑的处理层类 CarMessageService. Java，声明 updateCar（carMessage）。
- 对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 updateCarById。

【任务实施】

(1) 为“修改”按钮添加触发事件，在弹出层中显示要修改的汽车信息

① 为“修改”按钮添加触发事件，将请求发送到控制层。

- 在任务 7.4 中，输入车辆信息时，单击“修改”按钮触发相关事件，已经为“修改”按钮绑定了 onclick 事件 carupdate（vehID）。

```
html+="<div style='float: left;'><img alt='del' src='resources/images/edit.png'><a class=
\"btn btn-link editdeptbtn\" data-toggle=\"modal\" data-target=\"#modalupdate\" style='col-
or: #20c8ff' onclick=\"javascript:carupdate("+listdata[f].vehID+");\">修改</a></div>";
```

- 在 manage. jsp 页面中添加 carupdate()方法的定义，传递参数 vehID 车辆编号，将请求发送到 Control 层，代码如下。

```
1. function carupdate(id){
2.             let ids=id.id;
3.         $.ajax({
4.                 type:"post",
5.                 url:'<%=path%>'+"/carmessage/detail",
6.                 dataType : "jsonp",
7.                 jsonp:"jsoncallback",
```

```
8.                  data:{
9.                      "vehID":ids,
10.                 },
11.                 success : function(data){
12.                     $("#vehID2").val(data.vehID);
13.                     $("#plateNumber2").val(data.plateNumber);
14.                     $("#brand2").val(data.brand);
15.                     $("#currentMileage2").val(data.currentMileage);
16.                     $("#produceDate2").val(formatDate(data.produceDate));
17.                     $("#userId2").val(data.username);
18.                     $("#remarks2").val(data.remarks);
19.                 }
20.                 });
21.             }
```

代码说明：

- 第 2 行：获取车牌号。
- 第 4 行~第 10 行：设置 Ajax 调用参数，设置发送方式为 post，发送请求到控制层 detail，数据传递格式为 json 格式，传递参数车牌编号 ids。
- 第 11 行~第 19 行：控制层执行成功则返回数据 data，将 data 显示到指定对象中。

② 在 CarMessageController 中创建汽车信息类 carDetailBean，便于控制层到 View 层数据传递。

```
1. public static class carDetailBean {
2.         private String vehID;
3.         private String plateNumber;
4.         private String brand;
5.         private String remarks;
6.         private long produceDate;
7.         private Integer currentMileage;
8.         private String username;
9.         public String getVehID() {
10.            return vehID;
11.        }
12.    public void setVehID(String vehID) {
13.            this.vehID = vehID;
14.        }
15.    public String getPlateNumber() {
16.            return plateNumber;
17.        }
18.    public void setPlateNumber(String plateNumber) {
```

```
19.                 this. plateNumber = plateNumber;
20.             }
21.   public String getBrand( ) {
22.                 return brand;
23.             }
24.   public void setBrand( String brand) {
25.                 this. brand = brand;
26.             }
27. public String getRemarks( ) {
28.                 return remarks;
29.             }
30. public void setRemarks( String remarks) {
31.                 this. remarks = remarks;
32.             }
33. public long getProduceDate( ) {
34.                 return produceDate;
35.             }
36. public void setProduceDate( long produceDate) {
37.                 this. produceDate = produceDate;
38.             }
39. public Integer getCurrentMileage( ) {
40.                 return currentMileage;
41.             }
42. public void setCurrentMileage( Integer currentMileage) {
43.                 this. currentMileage = currentMileage;
44.             }
45. public String getUsername( ) {
46.                 return username;
47.             }
48. public void setUsername( String username) {
49.                 this. username = username;
50.             }     }
```

③ 在 CarMessageController 中创建 carDetail（vehID）方法，可根据指定车辆的车牌号获取车辆的详细信息。

```
1. @ RequestMapping( "/detail" )
2.     public void carDetail( HttpServletRequest req, HttpServletResponse resp)
3.             throws IOException {
4.         String vehID = req. getParameter( "vehID" );
5.         List<CarMessage> carMessages = carMessageService. carDetail( vehID);
6.         CarMessage carMessage = new CarMessage( );
7.         if ( carMessages. size( ) > 0 && carMessages ! = null) {
8.             carMessage = carMessages. get( 0);
```

```
9.          }
10.         User user = userService.getUserById(carMessage.getUserId());
11.         carDetailBean bean = new carDetailBean();
12.         bean.setVehID(carMessage.getVehID());
13.         bean.setBrand(carMessage.getBrand());
14.         bean.setCurrentMileage(carMessage.getCurrentMileage());
15.         bean.setPlateNumber(carMessage.getPlateNumber());
16.         bean.setRemarks(carMessage.getRemarks());
17.         if (user != null) {
18.             bean.setUsername(user.getUsername());
19.         }
20.         if (carMessage.getProduceDate() != null) {
21.             bean.setProduceDate(carMessage.getProduceDate().getTime());
22.         }
23.         String result = new Gson().toJson(bean);
24.         String jsonp = req.getParameter("jsoncallback");
25.         resp.setCharacterEncoding("UTF-8");
26.         resp.setContentType("text/html");
27.         if (jsonp != null) {
28.             result = jsonp + "(" + result + ")";
29.             resp.getWriter().write(result);
30.         } else {
31.             resp.getWriter().write(result);
32.         }
33.     }
```

代码说明：

- 第 4 行：获取车牌号
- 第 5 行：调用 carMessageService 服务的 carDetail（）方法，根据车牌获取车辆信息。
- 第 6 行~第 9 行：获取查得的车辆信息并存放在创建对象 carMessage 中。
- 第 10 行：通过 userService 的 getUserById（）方法获取用户信息。
- 第 11 行~第 22 行：将取得的车辆信息封装到 carDetailBean 对象 bean 中。
- 第 23 行~第 33 行：将数据转换为 json 格式，返回 View 层。

④ 在业务逻辑的处理层类完成相关处理。

- 在业务逻辑的处理层类 CarMessageService.Java，声明 carDetail 方法。

```
1. @Service
2. public class CarMessageService {
3.     @Autowired
4. private CarMessageMapper carMessageMapper;
5.     public List<CarMessage> carDetail(String vehId) {
```

```
6.              List<CarMessage> carMessages = carMessageMapper.detail(vehId);
7.              return carMessages;
8.      }
```

- 在业务逻辑的处理层类 userService.Java，声明 getUserById 方法。

```
1. @Service
2. public class UserService {
3.      @Autowired
4.      private UserMapper userMapper;
5. public User getUserById(String userid){
6.              return userMapper.getUserById(userid);
7.      }
```

⑤ 对象持久化映射层。

- 在 CarMessageMapper.java、CarMessageMapper.xml 中，声明和定义方法 detail(vehId)。

CarMessageMapper.java 代码如下：

```
public interface CarMessageMapper {
public List<CarMessage> detail(String vehId);
}
```

CarMessageMapper.java 中代码如下：

```
<select id="detail" parameterType="java.lang.String" resultType="com.piesat.zyms.domain.
core.CarMessage">
            select * from
            carmessage where
            vehID =#{vehID}
        </select>
```

- 在 UserMapper.java、UserMapper.xml 中，声明 getUserById 方法。

UserMapper.java 中代码如下：

```
1. public interface UserMapper {
2.      public User getUserById(String userid);
3. }
```

在 UserMapper.xml 中添加如下代码：

```
<select id="getUserById" parameterType="java.lang.String" resultType="com.
piesat.zyms.domain.core.User">
            select * from user where id = #{id}
        </select>
```

(2) 将修改过的汽车信息更新到数据库中

① 为“提交”按钮添加触发事件 changeCar()，将用户数据提交到控制层。

```
function changeCar(){
        $.ajax({
            type:"post",
            url:'<%=path%>'+"/carmessage/update",
            dataType : "jsonp",
            jsonp:"jsoncallback",
            data:{
            "vehID",$("#vehID2").val(),
            "plateNumber",$("#plateNumber2").val(),
            "brand",$("#brand2").val(),
            "currentMileage",$("#currentMileage2").val(),
            "produceDate",$("#produceDate2").val(),
            "userId",$("#userId2 option:selected").val(),
            "remarks",$("#remarks2").val(),
            },
            success : function(data){
                alert("更新成功");
            }
            });
    }
```

代码说明：

- 通过 Ajax 方法将用户修改过的车辆信息（vehID、plateNumber、brand、currentMileage、produceDate、userId、remarks）传递到控制层；如果执行成功，则显示更新成功。

② 在 CarMessageController 中创建修改信息方法 updateCar()。

```
@RequestMapping("update")
    public void updateCar(HttpServletRequest req, HttpServletResponse resp)
            throws IOException {
        // TODO 获取当前登录用户的 ID
        HttpSession session=req.getSession();
        String createId=(String) session.getAttribute("userId");
        String vehID = req.getParameter("vehID");
        String plateNumber = req.getParameter("plateNumber");
        String userId = req.getParameter("userId");
        String brand = req.getParameter("brand");
        Integer currentMileage = Integer.parseInt(req
                .getParameter("currentMileage"));
        String date = req.getParameter("produceDate");
        Date produceDate = null;
        DateFormat format = new SimpleDateFormat("yyyy-MM-dd");
```

```
16.             try {
17.                 produceDate = format.parse(date);
18.             } catch (Exception e) {
19.
20.             }
21.             String remarks = req.getParameter("remarks");
22.             CarMessage carMessage = new CarMessage();
23.             carMessage.setVehID(vehID);
24.             carMessage.setBrand(brand);
25.             carMessage.setCurrentMileage(currentMileage);
26.             carMessage.setCreateId(createId);
27.             carMessage.setPlateNumber(plateNumber);
28.             carMessage.setProduceDate(produceDate);
29.             carMessage.setRemarks(remarks);
30.             carMessage.setUserId(userId);
31.             carMessageService.updateCar(carMessage);
32.             String result = new Gson().toJson("OK");
33.             String jsonp = req.getParameter("jsoncallback");
34.             resp.setCharacterEncoding("UTF-8");
35.             resp.setContentType("text/html");
36.             if (jsonp != null) {
37.                 result = jsonp + "(" + result + ")";
38.                 resp.getWriter().write(result);
39.             } else {
40.                 resp.getWriter().write(result);
41.             }
42.     }
```

代码说明：

- 第 5 行~第 21 行：从 View 层获取车辆各字段的信息。
- 第 22 行~第 30 行：将获取的车辆信息封装到 carMessage 对象中。
- 第 31 行：调用 carMessageService 的 updateCar（）方法完成更新。
- 第 32 行~第 40 行：将执行结果以 json 格式输出返回。

③ 创建业务逻辑的处理层类 CarMessageService.Java，声明 updateCar（carMessage）。

```
1. @Service
2. public class CarMessageService {
3.     @Autowired
4.     private CarMessageMapper carMessageMapper;
5. public void updateCar(CarMessage carMessage){
```

```
6.              carMessageMapper. updateCarById( carMessage) ;
7.          }
8. }
```

④ 对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 updateCarById。

CarMessageMapper. java 中代码如下：

```
1. public interface CarMessageMapper{
2. public void updateCarById( CarMessage carMessage) ;
3. }
```

在 CarMessageMapper. xml 中添加如下代码：

```
1. <update id="updateCarById" parameterType="com. piesat. zyms. domain. core. CarMessage">
2.              update carmessage
3. <set>
4. vehID = #{vehId} ,
5.                  plateNumber = #{plateNumber} ,
6.                  brand = #{brand} ,
7.                  remarks=#{remarks} ,
8.                  producedate = #{produceDate} ,
9.                  currentmileage =
10.                 #{currentMileage} ,
11.                 userid=#{userId} ,
12.                 createid=#{createId}
13. </set>
14.             where vehId = #{vehId}
15.         </update>
```

代码说明：

- 本方法中要完成数据的修改，因此使用 update，同时修改指定车牌对应的车辆信息，因此修改的条件为车牌编号。

任务 7.6　解除绑定个人车辆信息

【任务描述】

通过 Spring+SpringMVC+MyBatis 框架实现个人车辆信息，用户登录后，通过单击“我的车辆”按钮进入车辆详情页面，可进行车辆的解绑，效果如图 7-6-1 所示。

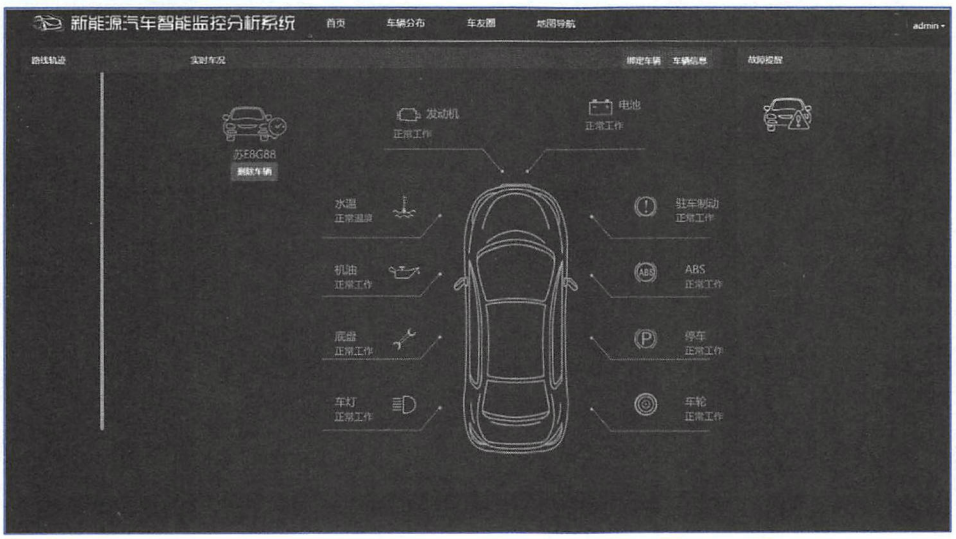


图 7-6-1　解绑车辆信息

【任务目标】

知识目标

- 熟悉 MyBatis 框架架构和原理。
- 熟悉 Spring+SpringMVC+MyBatis 框架的开发方法。

技能目标

- 使用 Spring+SpringMVC+MyBatis 框架并进行数据添加与删除。

【任务分析】

解除绑定流程是，单击“删除车辆”按钮，即可解绑。具体的开发流程如下。

① 为“删除车辆”按钮添加触发事件，将请求发送到控制层。

② 在 CarMessageController 中创建 deleteByUserid（）方法，可根据用户 ID 删除车辆信息。

③ 在业务逻辑的处理层类 CarMessageService. Java，声明 deleteByUserid（userId）方法。

④ 对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 deleteByUserid。

【任务实施】

① 在 MyselfCenter. jsp 页面中为“删除车辆”按钮添加触发事件，并请求发送到控制层。

```
<button class="btn btn-danger" onclick="del(${user.id})">删除车辆</button>
```

② 添加 onclick 事件。

```
1. function del(id){
2.         var id = id;
3.         $.ajax({
4.                 type : "post",
5.                 url : '<%=path%>' + "/carmessage/deleteByUserid",
6.                 data : {
7.                     userId : id
8.                 },
9.                 dataType : "json",              //数据类型为 json
10.                jsonp:"jsoncallback",
11.                success : function(data){
12.                if(data==="true"){
13.                        alert("删除成功!");
14.    window.location.href="http://" + location.host + '<%=path%>/' +'user/myself-
       center';
15.                }else{
16.                        alert("删除失败!");
17.            window.location.href="http://" + location.host + '<%=path%>/' + 'user/
       myselfcenter';
18.                }
19.            }
20.        });
21.    }
```

代码说明：

- 第 2 行：获取用户编号 id。
- 第 3 行~第 10 行：设置 Ajax 调用参数，设置发送方式为 post，发送请求到控制层 carmessage/deleteByUserid，数据传递格式为 json 格式，传递参数为用户编号 id。
- 第 11 行~第 19 行：控制层执行成功则返回数据 data，输出删除成功，否则显示删除失败。

③ 添加控制层 CarMessageController 的 deleteByUserid。

```
1. @RequestMapping("/deleteByUserid")
2.     public void deleteByUserid(HttpServletRequest req,
3.             HttpServletResponse resp, String userId) throws IOException {
4.         carMessageService.deleteByUserid(userId);
5.         CarMessage cm = carMessageService.getCarMessageByUser(userId);
6.         String result = "";
7.         String jsonp = "";
8.         if (cm == null) {
```

```
9.                result = new Gson( ). toJson( "true" ) ;
10.               jsonp = req. getParameter( "jsoncallback" ) ;
11.               resp. setCharacterEncoding( "UTF-8" ) ;
12.               resp. setContentType( "text/html" ) ;
13.               if (jsonp ! = null) {
14.                    result = jsonp + "(" + result + ")" ;
15.                    resp. getWriter( ). write( result) ;
16.               } else {
17.                    resp. getWriter( ). write( result) ;
18.               }
19.          } else {
20.               result = new Gson( ). toJson( "false" ) ;
21.               jsonp = req. getParameter( "jsoncallback" ) ;
22.               resp. setCharacterEncoding( "UTF-8" ) ;
23.               resp. setContentType( "text/html" ) ;
24.               if (jsonp ! = null) {
25.                    result = jsonp + "(" + result + ")" ;
26.                    resp. getWriter( ). write( result) ;
27.               } else {
28.                    resp. getWriter( ). write( result) ;
29.               }
30.          }
31.
32.     }
```

代码说明：

- 第 4 行：调用 carMessageService 服务的 deleteByUserid ( ) 方法，根据用户编号获取车辆信息。
- 第 5 行：根据用户编号查找车辆信息。
- 第 6 行~第 29 行：将结果回送到前台输出。

④ 实现业务逻辑层中 carMessageService 的 deleteByUserid( )方法。

```
1. public void deleteByUserid( String userId) {
2.          carMessageMapper. deleteByUserid( userId) ;
3.     }
```

⑤ 实现对象持久化映射层 CarMessageMapper. java、CarMessageMapper. xml。

CarMessageMapper. java、CarMessageMapper. xml，声明和定义方法 deleteByUserid(userId) 。

CarMessageMapper. java 代码如下：

```
public void deleteByUserid( String userId) ;
```

CarMessageMapper. java 代码如下：

```
<delete id="deleteByUserid" parameterType="java.lang.String">
        delete from carmessage
        where userid = #{userId}
</delete>
```

技能训练

1. 车辆信息查询

任务描述

基于 Spring+SpringMVC+MyBatis 框架并且结合 Ajax 技术实现后台管理员车辆查询，用户输入车辆查询关键字，可显示检索到的车辆信息。

任务分析

车辆查询中，需要根据用户输入的关键字进行检索，因此关键字将作为查询的条件传入到控制层，控制层进行处理后将查询到的数据在 manage.jsp 页面中显示。

2. 车辆信息删除

任务描述

基于 Spring+SpringMVC+MyBatis 框架并且结合 Ajax 技术实现指定车辆信息的删除。

任务分析

车辆信息删除时，需要将要删除的车牌号传递至控制层，在控制层调用后台服务处理后，将结果反馈给前台显示是否成功。该功能的实现可参照车辆信息更新，注意本功能中对数据库中要进行的是 delete 操作。

项目总结

在本项目中通过对车辆信息管理模块的实现，重点阐述了以下几项任务技能。

- Spring+SpringMVC+MyBatis 框架的功能开发。
- jqPaginator 分页插件的下载及使用。
- 使用 EL 和 JSTL 实现数据的显示。

以上这些技术基本涵盖了整个系统开发所用到的技术，但是本项目中车辆详情的显示业务逻辑比较复杂，不仅要了解框架技术的使用，还要理解项目的业务逻辑，具有一定的难度。

项目 8　车辆信息监控模块

PPT 车辆信息监控模块

项目描述

本项目将基于 Spring MVC+MyBatis 框架，在开发中应用数据可视化技术，把繁杂的车辆运行数据以图形图表的形式展现，为用户提供更直观形象的信息内容，更快速、精准地反映状态的变化和数据间的关联。 本项目通过车辆监控模块的开发，讲解 SpringMVC+MyBatis 框架下结合 Echarts 图表、百度地图等实现数据可视化，具体功能包括车辆信息监控、故障车详情、车辆统计、百度地图导航等。

知识目标

- 了解 Echarts 数据可视化图表开发组件。
- 了解百度地图开发组件。
- 熟悉 Bootstrap3 框架快速开发页面。
- 熟悉 SpringMVC 框架的运行原理。
- 熟悉 MyBatis 框架的运行原理。
- 熟悉基于 SpringMVC+MyBatis 框架的开发方法。

技能目标

- 能熟练通过 Bootstrap3 框架实现简单的网页开发。
- 能根据需求，熟练应用 SpringMVC+MyBatis 框架实现功能。

- 能使用实现基于 SpringMVC+MyBatis 框架的开发。
- 能使用 Echarts 等第三方组件初步实现数据的可视化图表展示。

任务列表

任务编号	任务名称	建议课时
任务 8.1	展示车辆状态监控地图	4
任务 8.2	实现按条件查询车辆信息	4
任务 8.3	展示故障车辆详情	6
任务 8.4	实现按省份统计分析车辆信息	2
任务 8.5	实现百度地图导航	课后
技能训练	制作首页车辆分布展示效果	2
	根据车牌号查询指定车辆详细信息	2
	总计：	20 课时

任务 8.1　展示车辆状态监控地图

【任务描述】

本任务采用百度的 Echarts 图表组件，制作车辆状态监控地图，实现不同车况的新能源汽车所在位置监控情况的可视化展示。

【任务目标】

知识目标

- 了解可视化图表组件 Echarts。
- 了解 Echarts 地图的配置和数据的显示方法

技能目标

- 能够初步利用 Echarts 实现地图中车辆位置展示。

微课 8-1
展示车辆状态监控地图

【任务分析】

ECharts 是百度公司的一个开源 javascript 图表库。它基于 HTML 5，兼容性好。通过 ECharts，可以呈现如折线图、柱状图、地图等多种类型的图表，方便实现数据可视化。根据开发需求，可以在 ECharts 官网找到案例模板，根据 API 及文档说明进行数据配置，便可呈现出炫酷的可视化界面。

【任务实施】

（1）创建车辆位置数据对应 POJO 类 CarLocation. java

在 src 文件夹的 com. piesat. zyms. domain. core 包中创建类 CarLocation 和 FaultInfo，类的属性如图 8-1-1 和图 8-1-2 所示，自动生成 getters and setters。

```
public class CarLocation {
    private Integer id;
    private String vehID;
    private String modID;
    private String realtime;
    private String latitudes;
    private String longitudes;
    private String remarks;
    private Date remarktime;
    private int count;
    private String province;
    private String plateNumber;
```

图 8-1-1　CarLocation 类属性

```
public class FaultInfo {
    private Integer id;
    private String faultID;
    private String faultState;
    private String faultLevel;
    private String vehID;
    private String modID;
    private String realtime;
    private String remarks;
    private Faultmean faultmean;
    private FaultSolution faultsolution;
    private int fid;
    private int count;
    private String fmodID;
    private String fremarks;
```

图 8-1-2　FaultInfo 类属性

（2）实现控制层 IndexController. java 中的 getFaultLocation( ) 方法

在 com. piesat. zyms. web. cms 包的 IndexController 中，创建 getFaultLocation 方法，该方法将返回车辆位置相关数据列表，代码如下。

```
package com.piesat.zyms.web.cms;
@RequestMapping("/energy")
@Controller
public class IndexController {
    @Autowired
    private FaultInfoService faultInfoService;

    @Autowired
    private CarLocationService carLocationService;
    ……
@RequestMapping(value = "/getfaultlocation")
public void getFaultLocation(HttpServletRequest req,HttpServletResponse resp)
            throws IOException,ServletException, ParseException {
        Map<String,Object> mapin = new HashMap<String,Object>();
        List<CarLocation> faultlocation = null;
        List<CarLocation> normallocation = null;
        faultlocation = carLocationService.getFaultCarLocation(mapin);
        normallocation = carLocationService.getNormalCarLocation(mapin);
        Map<String,Object> map = new HashMap<String,Object>();
        map.put("faultlocation", faultlocation);
        map.put("normallocation", normallocation);
        String result=new Gson().toJson(map);
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp != null){
            result = jsonp+"("+result+")";
            resp.getWriter().write(result);
        }else{
            resp.getWriter().write(result);
        }
    }
```

（3）实现业务逻辑的处理层类 CarLocationService.java

在 src 文件夹的包 com.piesat.zyms.service.cms 中新建类 CarLocationService，在该类中创建方法 getFaultCarLocation（）和 getNormalCarLocation（），获取故障车辆和正常运行车辆位置，代码如下。

```
package com.piesat.zyms.service.cms;
    @Service
    public class CarLocationService {

        @Autowired
        public class CarLocationService {
```

```
        private CarLocationMapper carLocationMapper;

    public List<CarLocation> getFaultCarLocation(Map<String,Object> map){
            return carLocationMapper.getFaultCarLocation(map);
        }

    public List<CarLocation> getNormalCarLocation(Map<String,Object> map){
            return carLocationMapper.getNormalCarLocation(map);
        }
}
```

(4) 实现对象持久化映射层 CarLocationMapper.java、CarLocationMapper.xml 和 FaultInfoMapper.java、FaultInfoMapper.xml

① 在包 com.piesat.zyms.persistence 中新建接口 CarLocationMapper.java，定义方法 getFaultCarLocation()和 getNormalCarLocation()，代码如下。

```
public interface CarLocationMapper{
    ……
    public List<CarLocation> getFaultCarLocation(Map<String,Object> map);
    public List<CarLocation> getNormalCarLocation(Map<String,Object> map);
}
```

② 在同一包中新建文件 CarLocationMapper.xml，在文件中定义以上两个方法的数据库操作，代码如下。

```
<mapper namespace="com.piesat.zyms.persistence.CarLocationMapper">
        <resultMap id="BaseResultMap" type="com.piesat.zyms.domain.core.CarLocation">
            <id column="id" property="id" />
            <result column="vehID" property="vehID" />
            <result column="modID" property="modID" />
            <result column="realtime" property="realtime" />
            <result column="latitudes" property="latitudes" />
            <result column="longitudes" property="longitudes" />
            <result column="remarks" property="remarks" />
            <result column="remarktime" property="remarktime" />
            <result column="count" property="count" />
            <result column="provid" property="provid" />
            <result column="plateNumber" property="plateNumber" />
        </resultMap>
        <select id="getFaultCarLocation" parameterType="java.util.Map" resultType=
"com.piesat.zyms.domain.core.CarLocation">
            select * from
            (select * from (select * from tbvehlocation l order by l.recordtime desc) j
group by j.vehID ) l,
```

```
                (select m. * ,a. faultname,a. id as aid from carmessage m,faultinfo f,fault a
where m. vehID = f. vehID and a. faulttype = f. faultLevel group by m. vehID) k
                    where l. vehID = k. vehID
                </select>

                <select id="getNormalCarLocation" parameterType="java. util. Map" resultType
="com. piesat. zyms. domain. core. CarLocation">
                    select * from
                    (select * from (select l. * ,g. plateNumber from tbvehlocation l, carmes-
sage g where g. vehID=l. vehID order by l. recordtime desc) j
                    group by j. vehID ) l
                    where l. vehID not in
                    (select carmessage. vehID from carmessage,faultinfo where carmessage. vehID
= faultinfo. vehID group by carmessage. vehID)
                </select>
</mapper>
```

(5) 下载 ECharts 组件

在 ECharts 官网的下载页面（http：//echarts. baidu. com/download. html）下载最新的版本，如图 8-1-3 所示。官网有多个版本的 ECharts 供下载，开发阶段建议选择源代码版本，该版本包含了常见的警告和错误提示，它是一个 js 文件 echarts. js。

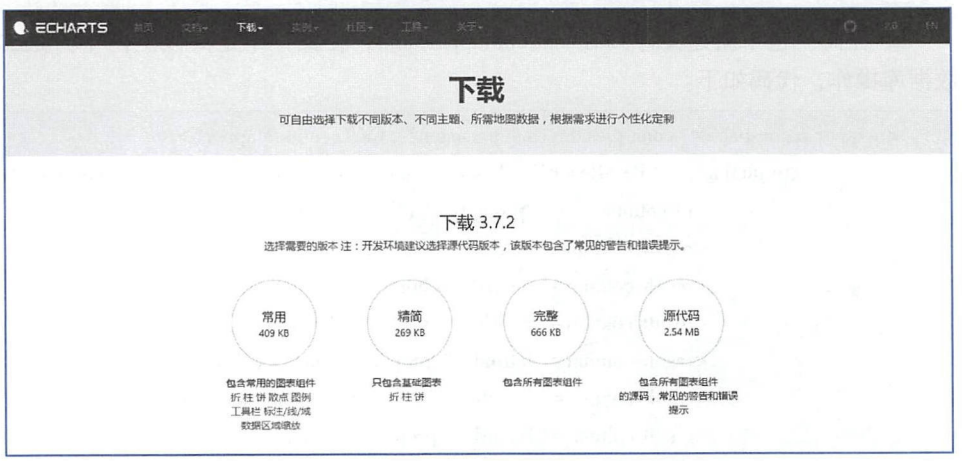


图 8-1-3　Echarts 下载

(6) 在页面中引入 Echarts

引入 Echarts 方式非常简单，只需要像普通的 JavaScript 库一样用 script 标签引入。例如下列代码。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```
<!--引入 ECharts 文件 -->
<script src="echarts.js"></script>
</head>
</html>
```

如果未下载 Echarts 组件，也可以通过 CDN 引入，推荐使用 BootCDN 上的库，如<script src=" https: //cdn. bootcss. com/echarts/3. 7. 2/echarts. js" ></script>。

(7) 在页面 carposition. jsp 中创建 ECharts 地图容器

在绘图前需要为 ECharts 提供一个具备高宽的 DOM 容器，这里创建一个名为 main 的层，代码如下。

```
<div id="main" style= "width: 1000px;height:750px"></div>
```

(8) 编写 js 代码，初始化 ECharts 实例

基于已有 DOM 容器，使用 init 方法初始化 ECharts 实例，代码如下。

```
<script>
var myChart = echarts. init(document. getElementById('main'));
</script>
```

(9) 编写 jQuery 代码，准备图表的配置项

本步骤为要生成的地图配置参数，也是最重要的部分，通过定制 option 参数，生成个性化的地图。配置项中有关车辆位置的数据常常通过 Ajax 技术异步调用数据库访问技术获取。编写的 ajax()方法中，将请求转发至 energy/getfaultlocation，返回的数据将作为地图配置的参数，最后，通过 setOption 方法，为 ECharts 实例图表指定配置项和数据，代码如下。

```
function pie(value,name) {
    this. value=value;
    this. name=name;
}
$. ajax({
    type : "post",
    url : '<%=path %>' + "/energy/getfaultlocation",
    data : { },
    dataType : "jsonp", //数据类型为 json
    jsonp:"jsoncallback",
    success : function(obj){
    var datas = [];
    var dataa = [];
    for(var i=0;i<obj. faultlocation. length;i++){
        var val = [obj. faultlocation[i]. latitudes,obj. faultlocation[i]. longitudes,1];
```

```
                myPie=new pie(val,obj.faultlocation[i].plateNumber);
                datas.push(myPie);
            }
            for(var i=0;i<obj.normallocation.length;i++){
                var val = [obj.normallocation[i].latitudes,obj.normallocation[i].longitudes,1];
                myPie=new pie(val,obj.normallocation[i].plateNumber);
                dataa.push(myPie);
            }
            datas=eval(datas);
            //地图参数配置
option = {
            backgroundColor: '#404a59',
            tooltip : { trigger: 'item'},
            bmap: {
        center: [118.78,32.04],
        zoom: 8,
        roam: true,
        mapStyle: {
        styleJson: [
                                {
                                    "featureType": "water",
                                    "elementType": "all",
                                    "stylers": {
                                        "color": "#044161"
                                    }
                                },
                                {
                                    "featureType": "land",
                                    "elementType": "all",
                                    "stylers": {
                                        "color": "#004981"
                                    }
                                },
                                {
                                    "featureType": "boundary",
                                    "elementType": "geometry",
                                    "stylers": {
                                        "color": "#064f85"
                                    }
                                },
                                {
                                    "featureType": "railway",
                                    "elementType": "all",
```

```
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "highway",
        "elementType" : "geometry",
        "stylers" : {
            "color" : "#004981"
        }
    },
    {
        "featureType" : "highway",
        "elementType" : "geometry. fill",
        "stylers" : {
            "color" : "#005b96",
            "lightness" : 1
        }
    },
    {
        "featureType" : "highway",
        "elementType" : "labels",
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "arterial",
        "elementType" : "geometry",
        "stylers" : {
            "color" : "#004981"
        }
    },
    {
        "featureType" : "arterial",
        "elementType" : "geometry. fill",
        "stylers" : {
            "color" : "#00508b"
        }
    },
    {
        "featureType" : "poi",
        "elementType" : "all",
```

```
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "green",
        "elementType" : "all",
        "stylers" : {
            "color" : "#056197",
            "visibility" : "off"
        }
    },
    {
        "featureType" : "subway",
        "elementType" : "all",
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "manmade",
        "elementType" : "all",
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "local",
        "elementType" : "all",
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "arterial",
        "elementType" : "labels",
        "stylers" : {
            "visibility" : "off"
        }
    },
    {
        "featureType" : "boundary",
        "elementType" : "geometry. fill",
```

```
                    "stylers" : {
                        "color" : "#029fd4"
                    }
                },
                {
                    "featureType" : "building",
                    "elementType" : "all",
                    "stylers" : {
                        "color" : "#1a5787"
                    }
                },
                {
                    "featureType" : "label",
                    "elementType" : "all",
                    "stylers" : {
                        "visibility" : "off"
                    }
                }
            ]
        }
    },
    series : [
        {
            name: '汽车位置',
            type: 'scatter',
            coordinateSystem: 'bmap',
            data: datas,
            symbolSize:5,
            label: {
                normal: {
                    formatter: '{b}',
                    position: 'right',
                    show: false
                },
                emphasis: {
                    show: true
                }
            },
            itemStyle: {
                normal: {
                    color: '#FF2222'
                }
            }
```

```
                },
                {
                    name: '汽车位置',
                    type: 'scatter',
                    coordinateSystem: 'bmap',
                    data: dataa,
                    symbolSize:5,
                    label: {
                        normal: {
                            formatter: '{b}',
                            position: 'right',
                            show: false
                        },
                        emphasis: {
                            show: true
                        }
                    },
                    itemStyle: {
                        normal: {
                            color: '#00DD2C'
                        }
                    }
                },
            ]
        };
        myChart.setOption(option);    //为 ECharts 实例图表指定配置项
    }
});
```

任务 8.2　实现按条件查询车辆信息

【任务描述】

本任务在任务 8.1 的基础上，实现车辆按条件查询功能，满足用户根据时间、地区及车况等不同条件监控车辆状态的需求，页面效果如图 8-2-1 所示。

图 8-2-1　按条件查询车辆

【任务目标】

知识目标

- 掌握 Echarts 地图的配置和数据的显示方法。

技能目标

- Spring MVC+MyBatis 框架下数据的读取及页面显示。
- 灵活使用 Echarts 实现地图中车辆位置展示。

微课 8-2
实现按条件查询车辆

【任务分析】

任务 8.1 已实现了所有车辆在地图中的位置显示，本任务将根据用户设置的条件进行查询，用返回的数据重新配置 ECharts 图表，展示出不同条件下车辆位置的分布状态。任务的实现可分成以下两步。

① 从数据库读取数据并在查询条件区域的“地区”及“车况”下拉列表中显示。

② 根据条件选项获取数据，为 ECharts 图表提供数据配置。

【任务实施】

(1) 读取数据并在下拉列表显示

① 在页面创建查询条件设置区域。

在页面 carposition. jsp 中，创建车辆查询条件区域，查询条件包括时间段、地区和车况等，HTML 代码如下。

```
<div class="leftBox">
<div class="lBoxHeader" style="font-size:10px;">
<div class="pull-left">车辆分布图</div>
<div class="pull-left">
<div class="form-group selectpo">
        <label for="name" style=" color: #59b2fb;font-weight: 500;">时间段</label>
            <input style="width:100px;font-size:12px;" type="text" id="startTime"
class="form-control Wdate" onfocus="WdatePicker({dateFmt:'yyyy-MM-dd'});"/>
            <label style="color:#59b2fb;">-</label>
            <input style="width:100px;font-size:12px;"  type="text" id="endTime"
class="form-control Wdate" onfocus="WdatePicker({dateFmt:'yyyy-MM-dd'});"/>
        </div>
    </div>
<div class="pull-left">
<div class="form-group selectpo">
        <label for="name" style=" color: #59b2fb;font-weight: 500;">地区</label>
<select class="form-control" style="width:100px;font-size:10px;" id="prov" >
    <option>全国</option>
</select></div>
```

```
</div>
<div class="pull-left">
        <div class="form-group selectpo">
            <label for="name" style=" color: #59b2fb;font-weight: 500;">车况</label>
            <select class="form-control"  id="carstatus">
                <option value="0">全部</option>
            </select>
            </div>
</div>
</div>
</div>
```

② 在控制层 IndexController. java 中实现 carposition( ) 方法。

在 com. piesat. zyms. web. cms 包的 IndexController 中，创建 carposition( ) 方法，该方法将返回全国省份地区相关数据列表和车辆状态列表，代码如下。

```
……
@ Autowired
private ChartService chartService;

@ RequestMapping(value = "/carposition", method = RequestMethod.GET)
    public ModelAndView carposition(HttpServletRequest request,
            HttpServletResponse response, ModelMap model) {
        Map<String,Object> map = new HashMap<String,Object>();
        List<Distribution> dis = chartService.getDistribution();
        List<Fault> fal = chartService.getFault();
        ModelAndView mv = new ModelAndView();
        mv.setViewName("car/carposition.jsp");
        mv.addObject("dis",dis);
        mv.addObject("fal",fal);
        return mv;
    }
```

③ 实现业务逻辑的处理层类 ChartService. java。

在 src 文件夹的包 com. piesat. zyms. service. cms 中新建类 ChartService，在该类中创建方法 getDistribution( ) 和 getFault( )，分别获取地区列表和车况列表，代码如下。

```
package com.piesat.zyms.service.cms;
@ Service
public class ChartService {
    @ Autowired
    private FaultMapper faultMapper;

    @ Autowired
```

```
    private DistributionMapper distributionMapper;

    public List<Fault> getFault( ) {
        return faultMapper. getFault( ) ;
    }
    public List<Distribution> getDistribution( ) {
        return distributionMapper. getDistribution( ) ;
    }
}
```

④ 实现对象持久化映射层 DistributionMapper. java、DistributionMapper. xml、FaultMapper. java、FaultMapper. xml。

● 在包 com. piesat. zyms. persistence 中新建接口 DistributionMapper. java，定义方法 getDistribution( )，代码如下。

```
public interface DistributionMapper {
    ……
    public List<Distribution>getDistribution( ) ;
}
```

新建接口 FaultMapper. java，定义方法 getFault( )，代码如下：

```
public interfaceFaultMapper{
    ……
    public List<Fault> getFault( ) ;
}
```

● 在同一个包中新建文件 DistributionMapper. xml、FaultMapper. xml，用于映射数据表 distribution、fault 的字段和表上的操作，代码分别如下。

```
<mapper namespace = " com. piesat. zyms. persistence. DistributionMapper" >
    <resultMap type = " com. piesat. zyms. domain. core. Distribution"  id = " distribution" >
        <id column = " ID"  property = " id" />
        <result column = " CARNUMBER"  property = " carnumber" />
        <result column = " LONGTUDE"  property = " longtude" />
        <result column = " LATITUDE"  property = " latitude" />
        <result column = " PROVINCE"  property = " province" />
        <result column = " COUNT"  property = " count" />
    <collection property = " carmessage"  ofType = " CarMessage" >
        <id column = " id"  property = " id"  />
        List<FaultInfo>fau = null;
        <result column = " brand"  property = " brand"  />
        <result column = " user_id"  property = " userId"  />
        <result column = " cartype"  property = " cartype"  />
    </collection>
    </resultMap>
```

```
    <select id="getDistribution" resultMap="distribution">
        select * from distribution
    </select>
</mapper><mapper namespace="com.piesat.zyms.persistence.FaultMapper">
    <resultMap type="com.piesat.zyms.domain.core.Fault" id="fault">
        <id column="ID" property="id"/>
        <result column="FAULTTYPE" property="faulttype"/>
        <result column="FAULTNUMBER" property="faultnumber"/>
        <result column="FAULTNAME" property="faultname"/>
        <result column="PROID" property="proid"/>
        <result column="COUNT" property="count"/>
    </resultMap>

    <select id="getFault" resultMap="fault">
        select * from fault
    </select>
</mapper>
```

⑤ 在视图层页面中显示数据。

通过 EL 表达式，将数据循环显示在页面表单中。

```
<label for="name" style=" color: #59b2fb;font-weight: 500;">地区:</label>
    <select class="form-control" style="width:100px;font-size:10px;" id="prov" >
        <option >全国</option>
        <c:forEach items="${dis}" var="var" varStatus="cs">
                <option value="${var.province}">${var.province}</option>
        </c:forEach>
    </select>
.....
<label for="name" style=" color: #59b2fb;font-weight: 500;">车况:</label>
        <select class="form-control" style="width: 100px; font-size: 10px;" id="carstatus">
            <option >全部</option>
            <c:forEach items="${fal}" var="var" varStatus="cs">
                <option value="${var.id}">${var.faultname}</option>
            </c:forEach>
        </select>
```

(2) 根据条件查询获取数据，配置 ECharts 图表数据

① 修改 IndexController 中 getFaultLocation( ) 方法。

修改 getFaultLocation( ) 方法，接收参数，根据参数获取相关数据。其中，变量 carstatus 表示车况，0 代表所有全部，1 代表正常运行车辆，其他为故障车辆。方法代码如下。

```
@RequestMapping(value = "/getfaultlocation")
    public void getFaultLocation(HttpServletRequest req,HttpServletResponse resp)
            throws IOException,ServletException, ParseException {
        String province = req.getParameter("province");          //省份
        String cars = req.getParameter("carstatus");             //车况
        String startTime = req.getParameter("startTime");        //时间
        String endTime = req.getParameter("endTime");
        int carstatus = 0;
        if(cars!=null){
            carstatus = Integer.parseInt(cars);
        }
        if("全国".equals(province)){
            province=null;
        }
        Map<String,Object> mapin = new HashMap<String,Object>();
        List<CarLocation> faultlocation = null;
        List<CarLocation> normallocation = null;
        mapin.put("province", province);
        mapin.put("startTime", startTime);
        mapin.put("endTime", endTime);
        if(carstatus!=0&&carstatus!=1){
            mapin.put("carstatus", carstatus);
            faultlocation = carLocationService.getFaultCarLocation(mapin);
            fau = faultInfoService.getfaultcount(mapin);
        }else if(carstatus==0){
            mapin.put("carstatus", "");
            faultlocation = carLocationService.getFaultCarLocation(mapin);
            normallocation = carLocationService.getNormalCarLocation(mapin);
            fau = faultInfoService.getfaultcount(mapin);
        }else if(carstatus==1){
            mapin.put("carstatus", 1);
            normallocation = carLocationService.getNormalCarLocation(mapin);
        }
        Map<String,Object> map = new HashMap<String,Object>();
        map.put("faultlocation", faultlocation);
        map.put("normallocation", normallocation);
        map.put("fau",fau);
        String result=new Gson().toJson(map);
        ……
    }
```

② 修改对象持久化映射层 CarLocationMapper.xml。

CarLocationMapper.xml 文件中获取故障车辆分布的 getFaultCarLocation 方法的原

有代码中添加查询条件，代码如下。

```
    < select  id = " getFaultCarLocation "  parameterType = " java. util. Map "  resultType = "
com. piesat. zyms. domain. core. CarLocation" >
            select * from
            ( select * from ( select * from tbvehlocation l order by l. recordtime desc ) j group
by j. vehID ) l,
            ( select m. * , a. faultname , a. id as aid from carmessage m , faultinfo f , fault a where
m. vehID = f. vehID and a. faulttype = f. faultLevel group by m. vehID) k
            where l. vehID = k. vehID
            <if test="startTime!= null and startTime!= " and endTime!= null and endTime!= "">
                and l. recordtime between #{ startTime , jdbcType = TIMESTAMP} and #{ end-
Time , jdbcType = TIMESTAMP}
            </if>
            <if test=" province!= null and province!= "">
                and  l. province = #{ province}
            </if>
            <if test=" carstatus!= null and carstatus!= "">
                and  k. aid = #{ carstatus}
            </if>
            <if test=" modid!= null and modid!= "">
                and  l. modID = #{ modid}
            </if>
        </select>
```

获取正常运行车辆分布的 getNormalCarLocation 方法的代码修改如下。

```
    <select id=" getNormalCarLocation"  parameterType=" java. util. Map"  resultType=" com.
piesat. zyms. domain. core. CarLocation" >
            select * from
            ( select * from ( select l. * , g. plateNumber from tbvehlocation l, carmessage g
where g. vehID=l. vehID order by l. recordtime desc) j
            group by j. vehID ) l
            where l. vehID not in
            ( select carmessage. vehID from carmessage , faultinfo where carmessage. vehID =
faultinfo. vehID group by carmessage. vehID)
            <if test="startTime!= null and startTime!= " and endTime!= null and endTime!= "">
                and l. recordtime between #{ startTime , jdbcType = TIMESTAMP} and #{ end-
Time , jdbcType = TIMESTAMP}
            </if>
            <if test=" province!= null and province!= "">
                and  l. province = #{ province}
            </if>
            <if test=" modid!= null and modid!= "">
```

```
                and   l.modID = #{modid}
            </if>
        </select>
```

③ 在视图层编写 jQuery 代码，为 ECharts 地图配置参数。

• 定义全国各省主要城市的地理坐标（经纬度），具体数值可通过网络查询得到。

```
var geoCoordMap = {
    '北京市':[116.46,39.92],
    '天津市':[117.2,39.13],
    '上海市':[121.48,31.22],
    '重庆市':[106.54,29.59]
    ……
};
```

• 编写 JavaSript 方法 drawchart()，该方法实现根据时间段、省份和车况查询数据并绘制图表，地图的配置项参见任务 8.1。drawchart()方法代码如下。

```
function drawchart() {
    var province = $("#prov").val();
    var position = "["+geoCoordMap[province]+"]";
    position=eval(position);
    var carstatus=$("#carstatus").val();
    var startTime = $("#startTime").val();
    var endTime = $("#endTime").val();
    $.ajax({
            type : "post",
            url : '<%=path %>' + "/energy/getfaultlocation",
            data : {
                    province:province,
                    carstatus:carstatus,
                    startTime:startTime,
                    endTime:endTime
            },
            dataType : "jsonp",//数据类型为 json
            jsonp:"jsoncallback",

        success : function(obj){
            var datas = [];
            var dataa = [];
            if(obj.faultlocation!=null){
                for(var i=0;i<obj.faultlocation.length;i++){
                    var val = [obj.faultlocation[i].latitudes,obj.faultlocation[i]
.longitudes,1];
```

```
                    myPie=new pie(val,obj.faultlocation[i].plateNumber);
                    datas.push(myPie);
                }
            }
            if(obj.normallocation!=null){
                for(var i=0;i<obj.normallocation.length;i++){
                    var val = [obj.normallocation[i].latitudes, obj.normallocation[i]
.longitudes,1];
                    myPie=new pie(val,obj.normallocation[i].plateNumber);
                    dataa.push(myPie);
                }
            }
            option={
        ......
                    series : [
                        {
                            name: '汽车位置',
                            type: 'scatter',
                            coordinateSystem: 'bmap',
                            data: datas,
                            symbolSize:5,
                            label: {
                                normal: {
                                    formatter: '{b}',
                                    position: 'right',
                                    show: false
                                },
                                emphasis: {
                                    show: true
                                }
                            },
                            itemStyle: {
                                normal: {
                                    color: '#FF2222'
                                }
                            }
                        },
                        {
                            name: '汽车位置',
                            type: 'scatter',
                            coordinateSystem: 'bmap',
                            data: dataa,
```

```
                symbolSize:5,
                label: {
                    normal: {
                        formatter: '{b}',
                        position: 'right',
                        show: false
                    },
                    emphasis: {
                        show: true
                    }
                },
                itemStyle: {
                    normal: {
                        color: '#00DD2C'
                    }
                }
            },
        ]
    };
    myChart.setOption(option);
    }
});
    }
```

- 调用 drawchart()方法，代码如下。

```
<!--省份列表 -->
<select class="form-control"  id="prov" onchange="drawchart()" >
        <option value="江苏省">全国</option>
        <c:forEach items="${dis}" var="var" varStatus="cs">
            <option value="${var.province}">${var.province}</option>
        </c:forEach>
    </select>
<!--车况列表 -->
<select class="form-control"  id="carstatus" onchange="drawchart()" >
    <option value="0">全部</option>
        <c:forEach items="${fal}" var="var" varStatus="cs">
            <option value="${var.id}">${var.faultname}</option>
        </c:forEach>
</select>
```

任务 8.3　展示故障车辆详情

【任务描述】

本任务将实现车辆故障列表及详情展示。在故障列表中，显示不同部件故障车辆的数量，通过“详情”超链接查看故障车辆详情及故障示意图，页面效果如图 8-3-1～图 8-3-3 所示。

微课 8-3
展示故障车辆详情

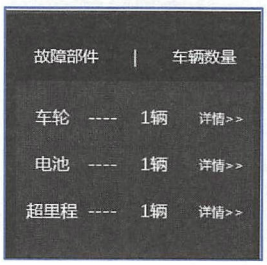


图 8-3-1　故障车辆统计列表

图 8-3-2　故障车辆详情

图 8-3-3　车辆故障示意图

【任务目标】

知识目标

- 熟练掌握 Spring MVC+MyBatis 框架下的数据读取及显示方法。

技能目标

- 能在 Spring MVC+MyBatis 框架下实现数据的读取及页面显示。

【任务分析】

(1) 故障车辆统计列表显示

- 在 IndexController 中实现方法，获取不同部件故障车辆数量。
- 在 faultInfoService. java 中修改 getfaultcount 方法，获取故障车故障部件及数量。
- 在 FaultInfoMapper. java、FaultInfoMapper. xml 中声明和定义 tbvehlocation 表、carmessage 表、faultinfo 表、fault 上的联合查询方法 getfaultcount 方法。
- 在视图层 carposition. jsp 页面中，通过循环输出故障车辆数量列表。

(2) 故障车辆详情显示

- 在 IndexController 中实现方法，根据故障类型返回故障车辆数据。
- 在 faultInfoService. java 中定义 getfaultinfoBymodID 方法，获取某种故障的车辆数据。
- 在 FaultInfoMapper. java、FaultInfoMapper. xml 中声明和定义 faultinfo 表、tbfaultsolution表、tbfaultmean 表上的联合查询方法 getfaultinfoBymodID 方法。
- 在视图层 carposition. jsp 页面中，设置超链接并提交至 IndexController，将返回的数据显示在视图层故障车辆详情页面 fault. jsp 中。

【任务实施】

1. 故障车辆统计列表显示

(1) 车辆故障信息数据对应 POJO 类 FaultInfo. java

该类在前一个项目中已完成创建，详见任务 7.4。

(2) 修改控制层 IndexController. java 中的 getFaultLocation( ) 方法

进一步修改 getFaultLocation 方法，添加获取故障车辆信息的操作，代码如下。

```
@RequestMapping(value = "/getfaultlocation")
public void getFaultLocation(HttpServletRequest req,HttpServletResponse resp)
        throws IOException,ServletException, ParseException {
……
    List<CarLocation> faultlocation = null;
    List<CarLocation> normallocation = null;
    List<FaultInfo> fau = null;
    ……
    if(carstatus!=0&&carstatus!=1){
```

```
            ……
                fau = faultInfoService.getfaultcount(mapin);
            }else if(carstatus==0){
                ……
                fau = faultInfoService.getfaultcount(mapin);
            }else if(carstatus==1){
                ……
            }
            ……
            map.put("fau", fau);
            String result=new Gson().toJson(map);
            ……
        }
```

(3) 在业务逻辑的处理层类 FaultInfoService.java 中创建方法

在包 com.piesat.zyms.service.cms 中的类 FaultInfoService 中创建方法 getFaultCount()，获取故障车辆相关数据，代码如下。

```
public class FaultInfoService {
    @Autowired
    private FaultInfoMapper faultInfoMapper;

    public List<FaultInfo> getfaultcount(Map<String,Object> map){
        return faultInfoMapper.getfaultcount(map);
    }
}
```

(4) 实现对象持久化映射层 FaultInfoMapper.java、FaultInfoMapper.xml

① 在包 com.piesat.zyms.persistence 接口文件 FaultInfoMapper.java 中定义方法 getfaultcount()，代码如下。

```
public interface FaultInfoMapper{
    ……
    public List<FaultInfo> getFaultCount(Map<String,Object> map);
}
```

② 在 FaultInfoMapper.xml 声明和定义 tbvehlocation 表、carmessage 表、faultinfo 表、fault 表上的联合查询方法 getFaultCount，代码如下。

```
<mapper namespace="com.piesat.zyms.persistence.FaultInfoMapper">
    <resultMap id="BaseResultMap" type="com.piesat.zyms.domain.core.FaultInfo">
        <id column="id" property="id" />
        <result column="vehID" property="vehID" />
        <result column="faultID" property="faultID" />
```

```
            <result column="faultState" property="faultState" />
            <result column="faultLevel" property="faultLevel" />
            <result column="modID" property="modID" />
            <result column="realtime" property="realtime" />
            <result column="remarks" property="remarks" />
            <result column="count" property="count" />
            <result column="fremarks" property="fremarks" />
            <result column="fmodID" property="fmodID" />
            <result column="fid" property="fid" />
        <association property="faultmean" javaType="Faultmean">
            <id column="id" property="id" />
            <result column="faultmean" property="faultmean"/>
        </association>
        <association property="faultsolution" javaType="FaultSolution">
            <id column="id" property="id" />
            <result column="faultSolution" property="faultSolution"/>
        </association>
    </resultMap>
    <select id="getFaultCount" parameterType="java.util.Map"
      resultType="com.piesat.zyms.domain.core.FaultInfo">
        select count(*) as count, k.fid, k.faultID, k.fmodID, k.fremarks, k.faultState,
k.fremarks from
            (select * from (select * from tbvehlocation l order by l.recordtime desc) j group
by j.vehID ) l,
            (select m.*,f.id as fid,f.faultID,f.faultState,f.modID as fmodID,f.remarks as
fremarks,a.faultname,a.id as aid from carmessage m,faultinfo f,fault a where m.vehID = f.vehID
and a.faulttype = f.faultLevel group by m.vehID) k
            where l.vehID = k.vehID
            group by fremarks
        </select>
    </mapper>
```

(5) 在视图层页面中显示数据

在 carposition.jsp 页面中添加 HTML 代码，并将返回的数据显示在页面中，代码如下：

```
    <div class="rBoxHeader">
    <div class="col-md-5">故障部件</div>
    <div class="col-md-2">|</div>
    <div class="col-md-5">车辆数量</div>
    </div>
    <div class="rBoxContent" style="height:250px;">
        <c:forEach items="${fau}" var="fau" varStatus="cs">
```

```
        <div class="row citylists" >
        <div class="pull-left pro">${fau.faultState}</div>
        <div class="pull-left lable">----------------</div>
        <div class="pull-left carnum">${fau.count}辆</div>
    </div>
    </c:forEach>
    </div>
```

2. 故障车辆详情显示

(1) 在控制层 IndexController.java 中的创建 fault() 方法

在 com.piesat.zyms.web.cms 包的 IndexController 中，创建 fault() 方法，该方法将根据故障类型返回故障车辆数据，代码如下。

```
@RequestMapping(value = "/fault/{type}", method = RequestMethod.GET)
public ModelAndView fault(HttpServletRequest request,
            HttpServletResponse response, ModelMap model,@PathVariable String type) {
    List<FaultInfo> faultinfo = faultInfoService.getfaultinfoBymodID(type);
    ModelAndView mv = new ModelAndView("car/fault.jsp","faultinfolist",faultinfo);
    return mv;
}
```

(2) 在业务逻辑的处理层类 FaultInfoService.java 中创建方法

在类 FaultInfoService 中创建方法 getfaultinfoBymodID()，根据故障类型获取故障车辆相关数据，代码如下。

```
@Service
public class FaultInfoService {
    @Autowired
    private FaultInfoMapper faultInfoMapper;

    ......
    public List<FaultInfo> getfaultinfoBymodID(String modID){
        return faultInfoMapper.getfaultinfoBymodID(modID);
    }
    ......
}
```

(3) 在已有对象持久化映射层 FaultInfoMapper.java、FaultInfoMapper.xml 添加方法

① 在已创建的 FaultInfoMapper 接口中添加方法 getfaultinfoBymodID()，代码如下。

```
public interface FaultInfoMapper{
    ......
    public List<FaultInfo> getFaultCount(Map<String,Object> map);
```

```
    public List<FaultInfo> getfaultinfoBymodID(String modID);
}
```

② 在 FaultInfoMapper.xml 中声明和定义 faultinfo 表、tbfaultsolution 表、tbfaultmean 表上的联合查询方法 getfaultinfoBymodID，代码如下。

```
<select id="getfaultinfoBymodID"    parameterType="java.lang.String"
resultMap="BaseResultMap">
    select * from faultinfo,tbfaultsolution,tbfaultmean where
    faultinfo.modID=#{modID} and faultinfo.faultID=tbfaultsolution.faultID and
    faultinfo.faultID=tbfaultmean.faultID
</select>
```

(4) 在视图层添加超链接并显示数据

在视图层 carposition.jsp 页面的故障车辆统计列表中，添加“详情”超链接，代码如下。

```
<c:forEach items="${fau}" var="fau" varStatus="cs">
    <div class="row citylists" >
    ……
    <div class="pull-left cardetail">
            <a href="${request.contextPath}/car/energy/fault/${fau.fmodID}">
                详情>>
            </a>
        </div>
    </div>
</c:forEach>
```

在故障车辆详情页面 fault.jsp 中，读取方法返回的数据，并显示在页面中，代码如下。

```
<section id="content">
    <div class="col-md-12" style="color:#fff;background-color:#196BA2;">
        <div class="col-md-6 pull-left">
        <h1 class="pull-left" style="margin-bottom: 10px">
        <span style="font-size: 20px">
        <c:forEach items="${faultinfolist}" var="faultinfolist" varStatus="fa">
            <c:if test="${faultinfolist.modID == 1}">发动机故障</c:if>
            <c:if test="${faultinfolist.modID == 2}">轮胎故障</c:if>
            <c:if test="${faultinfolist.modID == 3}">电池故障</c:if>
          <c:if test="${faultinfolist.modID == 4}">超里程故障</c:if>
        </c:forEach>
        </span>
        </h1>
    </div>
```

```
        <div class="clearfix"></div>
        </div>
        <div class="col-md-12" style="color: #fff ;margin-top: 10px;">
            <table id="user-table" class="table">
                <thead>
                    <tr>
                    <th style="text-align:left">车号</th>
                    <th style="text-align:left">故障名称</th>
                    <th style="text-align:left">故障级别</th>
                    <th style="text-align:left">解决方法</th>
                    <th style="text-align:left">操作</th>
                    </tr>
                </thead>
                <tbody>
                <c:forEach items="${faultinfolist}" var="faultinfolist" varStatus="fa">
                    <tr>
                    <td>${faultinfolist.vehID}</td>
                <td>${faultinfolist.faultmean.faultmean}</td>
                    <td>${faultinfolist.faultLevel}</td>
                <td>${faultinfolist.faultSolution.faultSolution}</td>
                    <td>
                <div style="float: left;">
                    <img alt="del" src="resources/images/view.png">
                        <a style="color: #20c8ff" href="javascript:void(0)"
                                onclick="view('${faultinfolist.vehID}')">查看</a>
                </div>
                </td>
                </tr>
                </c:forEach>
                </tbody>
        </table>
        </div>
        <div style="text-align: center;margin-top: 20px;">
            <ul id="pagination"></ul>
        </div>
        </section>
```

上述代码中，“查看”按钮的 click 事件调用 view 方法，跳转至 CarInformation.jsp 页面，根据车辆编号显示车辆详情，View 方法的代码如下。

```
function view(vehID) {
                    var id = encodeURI(encodeURI(vehID));
                    window.location.href='http://' + location.host + '<%=path %>/'
```

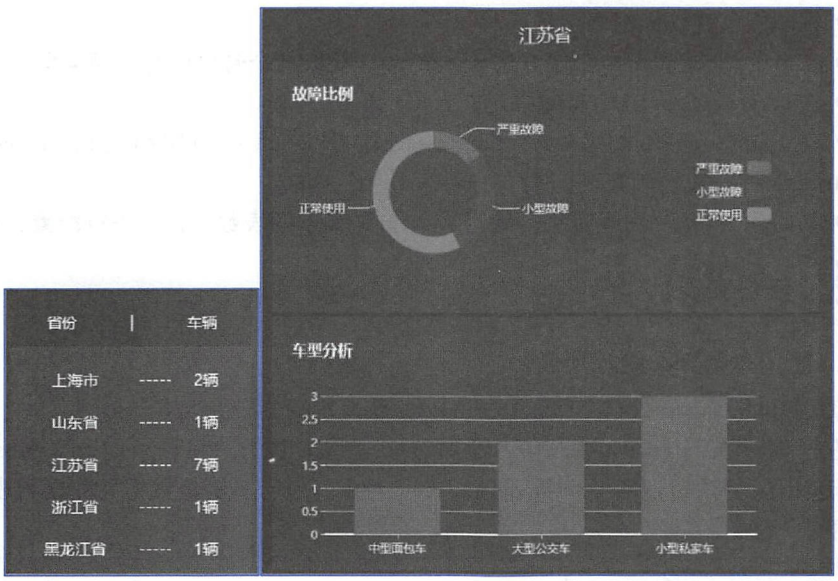
```
                +'energy/carinformation/'+id;
    }
```

在视图层 carinformation. jsp 页面中显示数据，其功能已在任务 7.4 中实现。

任务 8.4　实现按省份统计分析车辆信息

【任务描述】

本任务将实现各省份车辆分布统计列表，单击列表中的每一项，将显示该省不同车况车辆占比圆环图和各种车型数据的柱形图，页面效果如图 8-4-1 所示。



(a)　　(b)

图 8-4-1　各省车辆统计分析

【任务目标】

知识目标

- 了解并掌握 Echarts 柱状图、圆环图等的配置和显示。

技能目标

- Spring MVC+MyBatis 框架下数据的读取及页面显示的熟练运用。
- 能使用 Echarts 组件绘制柱状图、圆环图等统计图表。

微课 8-4
实现按省份统计分析车辆信息

【任务分析】

(1) 显示各省份车辆分布统计列表

- 在 IndexController 中实现方法，获取当前车辆分布列表。

- 在 ChartService. java 中定义 getCarDistri 方法，获取车辆分布列表数据。
- 在 ChartMapper. java、ChartMapper. xml 中声明和定义 cardistribution 表、distribution 表上的联合查询方法 getCarDistri 方法。
- 在视图层 index. jsp 页面中，通过循环输出车辆分布列表。

(2) 故障车辆占比圆环图显示

- 视图层创建圆环图表容器。
- 在 IndexController 中实现方法 getFault，获取某个省份正常行驶和故障车数量。
- 在 ChartService. java 中定义 getListById、getCountBydid 方法。
- 在 FaultMapper. java、FaultMapper. xml 中声明和定义 fault、faultinfo 等表上的联合查询方法 getListById、getCountBydid 方法。
- 初始化 ECharts 图表实例，编写 Ajax 方法请求图表数据，并进行配置、显示。

(3) 不同车型车辆数柱形图显示

- 视图层创建柱形图表容器。
- 在 IndexController 中实现方法 getcartype，获取某个省份所有车辆类型。
- 在 carMessageService. java 中定义 getCarDistri 方法。
- 在 CarMessageMapper. java、CarMessageMapper. xml 中声明和定义 carmessage、cardistribution 等表上的联合查询方法 getCarDistri 方法。
- 初始化 ECharts 图表实例，编写 Ajax 方法请求图表数据，并进行配置、显示。

【任务实现】

1. 显示车辆分布统计列表

① 在 IndexController 中实现方法 index，获取车辆分布列表。

在已创建的 IndexController 中，添加 index 方法，该方法将调用 ChartService 类的 getCarDistri( ) 方法，返回一个 ModelAndView 对象，包含车辆分布相关数据列表，代码如下。

```
package com. piesat. zyms. web. cms;
@ RequestMapping( "/energy" )
@ Controller
public class IndexController {
    @ Autowired
    private ChartService chartService;
    ……
    @ RequestMapping( value = "/index" , method = RequestMethod. GET)
    public ModelAndView index( HttpServletRequest request,
        HttpServletResponse response, ModelMap model) {
        List<Distribution> distribution = chartService. getCarDistri( );
        ModelAndView mv = new ModelAndView( );
        mv. addObject( "distribution" ,distribution) ;
        mv. setViewName( "index. jsp" ) ;
        return mv;
    }
```

在前面任务中，ChartService. java、ChartMapper. java、ChartMapper. xml 文件均已创建，getCarDistri( )也已存在，本任务中不再赘述。

② 在视图层 index. jsp 页面中输出各省车辆分布列表。

在 index. jsp 页面中，读取控制层 index 方法返回的 distribution 列表对象，通过循环在页面进行输出，代码如下。

```
<div class="rightBox">
<div class="rBoxHeader">
<div class="col-md-5">省份</div>
<div class="col-md-2">|</div>
<div class="col-md-5">车辆</div>
</div>

<div class="rBoxContent">
<c:forEach items="${distribution}" var="var" varStatus="vs">
    <div class="row citylists" >
    <div class="pull-left pro">${var.province}</div>
    <div class="pull-left lable">----------------</div>
    <div class="pull-left carnum">${var.count}辆</div>
</div>
</c:forEach>
</div>
</div>
```

2. 故障车辆占比圆环图显示

① 在视图层中创建圆环图的容器。在 index. jsp 页面中，添加 Bootstrap 模态框代码，在模态框中创建圆环图容器，代码如下。

```
<!--模态框(Modal)-->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="myModalLabel" aria-hidden="false">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-hidden="true">&times;</button>
                <h4 class="modal-title" id="myModalLabel"></h4>
            </div>
            <div class="modal-body">
                <div class="col-md-12 boxpro">
                <div id="chart" style="width:508px;height:250px;"></div>
```

```
                    </div>
                    <div class="clearfix"></div>
                </div>
            </div>
        </div>
    </div>
```

② 创建车辆故障对应 POJO 类 Fault.java。在 src 文件夹的 com.piesat.zyms.domain.core 包中创建类 Fault，类的属性如图 8-4-2 所示，自动生成 getters and setters。

```
public class Fault {

    private int id;
    private int faulttype;
    private int faultnumber;
    private String faultname;
    private int proid;
    private int count;
```

图 8-4-2　Fault 类的属性

③ 在 IndexController 类中创建 getFault 方法，该方法将获取某省份正常行驶和故障车辆的数量。

```
@RequestMapping(value = "/getfault")
    public void getFault(HttpServletRequest req,HttpServletResponse resp)
            throws IOException,ServletException, ParseException {
        int distributionid = Integer.parseInt(req.getParameter("id"));
        List<Fault> fault = chartService.getListById(distributionid);
        int faultcount = 0;
        for (Fault f : fault) {
            faultcount += f.getCount();
        }
        int total = chartService.getCountBydid(distributionid);
        int normalcount = total - faultcount;
        if(normalcount != 0){
            Fault norf = new Fault();
            norf.setCount(normalcount);
            norf.setFaultname("正常使用");
            fault.add(norf);
        }
        String result=new Gson().toJson(fault);
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp != null){
```

```
                result = jsonp+"("+result+")";
                resp.getWriter().write(result);
            }else{
                resp.getWriter().write(result);
            }
        }
```

④ 实现业务逻辑的处理层类 ChartService 中的 getListById、getCountBydid 方法。

```
package com.piesat.zyms.service.cms;
@Service
public class ChartService {

    @Autowired
    private FaultMapper faultMapper;

    public List<Fault> getListById(int id){
        return faultMapper.getListById(id);
    }

    public int getCountBydid(int id){
            return faultMapper.getCountBydid(id);
    }
}
```

⑤ 实现对象持久化映射层 FaultMapper.java、FaultMapper.xml。

- 在接口 FaultMapper.java 中，定义 getListById 和 getCountBydid 两个方法，代码如下。

```
public interface FaultMapper {
    ……
    public List<Fault> getListById(int id);
    public int getCountBydid(int distributionid);
}
```

- 在文件 FaultMapper.xml 中，添加 getListById 和 getCountBydid 的方法定义，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"" http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.piesat.zyms.persistence.FaultMapper">
    ……
    <select id="getListById" parameterType="java.lang.Integer" resultMap="fault">
```

```
            select count( * ) as count,f. * from fault f,faultinfo i,carmessage m,cardistribution
d where f. faulttype = i. faultLevel and m. vehID = i. vehID and d. carmessageid = m. id and
d. distributionid=#{distributionid} group by f. faultname
        </select>

        < select id = " getCountBydid " parameterType = " java. lang. Integer " resultType =
" java. lang. Integer" >
            select count( * ) from cardistribution d where d. distributionid = #{distributionid}
        </select>
    </mapper>
```

⑥ 初始化图表实例，编写 Ajax 方法请求数据，并进行配置。在 index. jsp 页面中，添加 drawChart 方法，该方法对图表进行初始化，并通过 Ajax 请求数据，对图表进行配置，方法代码如下。

```
function drawChart(id){
    var chart = echarts. init(document. getElementById('chart'));
    var type = [];
    var number = [];
    $. ajax({
            type : "post",
            async:false,
            url : '<%=path %>' + "/energy/getfault",
            data : {
                    id:id
            },
            dataType : "jsonp",
            jsonp:"jsoncallback",
            success : function(data){
                for(var i=0;i<data. length;i++){
                    myPie=new pie(data[i]. count,data[i]. faultname);
                    type. push(data[i]. faultname);
                    number. push(myPie);
                }
    optionchart = {
                title: {
                    text: '故障比例',
                    textStyle: {
                        color: '#fff',
                        fontSize:16
                    },
                    textAlign:'left'
                },
```

```
        tooltip: {
            trigger: 'item',
            formatter: "{a} <br/>{b}: {c} ({d}%)"
        },
        legend: {
            orient: 'vertical',
            x: 'right',
            bottom: '35%',
            textStyle:{
        color: '#fff',
                fontSize:12

            },
            data:type
        },
        series: [
            {
                name:'故障数量',
                type:'pie',
                radius: ['40%', '55%'],
                label: {
                    normal: {
                        textStyle: {
                            color: '#fff'
                        }
                    }
                },
                labelLine: {
                    normal: {
                        lineStyle: {
                            color: '#fff'
                        },
                        smooth: 0.2,
                        length: 10,
                        length2: 20
                    }
                },
                center: ['30%', '50%'],
                data:number
            }
        ]
    };
```

```
            chart.setOption(optionchart);
                    }
        });
    }
```

⑦ 完善视图层 index.jsp 页面的 HTML 代码，调用 jQurey 方法，显示图表。

- 编写方法 detail，该方法将弹出一个模态框，并在模态框中绘制图表，代码如下。

```
function detail(id,prov){
    $("#myModal").modal('show');
    $("#myModalLabel").html(prov);
    drawChart(id);
}
```

- 在 index.jsp 页面中 class 为 row citylists 的 div 中添加 onclick 事件属性，实现功能“显示车辆分布统计列表”，代码如下。

```
<div class="rBoxContent">
<c:forEach items="${distribution}" var="var" varStatus="vs">
    <div class="row citylists" onclick="detail('${var.id}','${var.province}')">
    <div class="pull-left pro">${var.province}</div>
    <div class="pull-left lable">----------------</div>
    <div class="pull-left carnum">${var.count}辆</div>
</div>
</c:forEach>
</div>
```

3. 不同车型车辆数柱形图显示

(1) 在视图层 index.jsp 中创建柱形图的容器

在“显示车辆分布统计列表”功能中创建的模态框中，继续创建柱形图 div 容器，代码如下。

```
<!-- 模态框(Modal)-->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby=
"myModalLabel" aria-hidden="false">
        ......
            <div class="modal-body">
                <div class="col-md-12 boxpro">
                <div id="chart" style="width:508px;height:250px;"></div>
                <div id="line" style="width:508px;height:250px;"></div>
                </div>
                <div class="clearfix"></div>
    </div>
</div>
```

（2）在 IndexController 中创建 getCarType 方法

在 IndexController 类中创建 getCarType 方法，代码如下。

```
@RequestMapping(value = "/getcartype")
    public void getCarType(HttpServletRequest req,HttpServletResponse resp)
            throws IOException,ServletException, ParseException {
        int distributionid = Integer.parseInt(req.getParameter("id"));
        List<CarMessage> carm = carMessageService.getCarDistri(distributionid);
        String result=new Gson().toJson(carm);
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp!= null){
            result = jsonp+"("+result+")";
            resp.getWriter().write(result);
        }else{
            resp.getWriter().write(result);
        }
    }
```

（3）实现业务逻辑的处理层类 carMessageService 中的 getCarDistri 方法

在类 carMessageService.java 中，创建 getCarDistri 方法，代码如下。

```
package com.piesat.zyms.service.cms;
@Service
public class carMessageService {

@Autowired
private CarMessageMapper carMessageMapper;

public List<CarMessage> getCarDistri(int distributionid){
        return carMessageMapper.getCarDistri(distributionid);
    }
}
```

（4）实现对象持久化映射层 CarMessageMapper.java、CarMessageMapper.xml

① 在接口 CarMessageMapper.java 中，定义 getCarDistri 方法，代码如下。

```
public interface CarMessageMapper {
    ……
    public List<CarMessage> getCarDistri(int distributionid);

}
```

② 在文件 CarMessageMapper.xml 中，添加 getCarDistri 的方法定义，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.piesat.zyms.persistence.FaultMapper">
    ……
    <select id="getCarDistri" parameterType="java.lang.Integer" resultMap="BaseResultMap">
        select count(*) as count, m.* from carmessage m , cardistribution c where c.carmessageid = m.id and c.distributionid=#{distributionid} group by m.cartype
    </select>
</mapper>
```

（5）初始化图表实例，编写 Ajax 方法请求数据，并进行配置

在 index.jsp 页面中，添加 drawLine()方法，与 drawChart 方法类似，该方法对图表进行初始化，并通过 Ajax 请求数据，配置图表，方法的代码如下。

```
function drawLine(id){
    var line = echarts.init(document.getElementById('line'));
    var type = [];
    var number = [];
    $.ajax({
            type : "post",
            async:false,
            url : '<%=path %>' + "/energy/getcartype",
            data : {
                id:id
            },
            dataType : "jsonp",                          //数据类型为 json
            jsonp:"jsoncallback",
            success : function(data){
                for(var i=0;i<data.length;i++){
                    type.push(data[i].cartype);
                    number.push(data[i].count);
                }
    optionline = {
            title: {
                text: '车型分析',
                textStyle: {
                    color: '#fff',
                    fontSize:16
                },
                textAlign:'left'
            },
```

```
color: ['#c23531','#2f4554', '#61a0a8'],
tooltip : {
    trigger: 'axis',
    axisPointer : {                    // 坐标轴指示器,坐标轴触发有效
        type : 'shadow'                // 默认为直线,可选为:'line' | 'shadow'
    }
},
grid: {
    left: '3%',
    right: '4%',
    bottom: '3%',
    containLabel: true
},
xAxis : [
    {
        type : 'category',
        axisLine:{
            lineStyle:{
                color:'#fff',
                width:2
            }
        },
        data : type,
        axisTick: {
            alignWithLabel: true
        }
    }
],
yAxis : [
    {
        type : 'value',
        axisLine:{
            lineStyle:{
                color:'#fff',
                width:0
            }
        },
    }
],
series : [
    {
        name:'数量',
```

```
                        type:'bar',
                        barWidth: '60%',
                        data:number,
                    }
                ]
            };
            line.setOption(optionline);
            }
        });
    }
```

(6) 在 detail 方法中，调用 drawChart 方法，显示图表

```
function detail(id,prov){
    ......
    drawChart(id);
    drawLine(id);
}
```

任务 8.5　实现百度地图导航

【任务描述】

通过调用百度地图 API 实现地图显示、搜索地址、查询路线等功能。

【任务目标】

微课 8-5
实现百度地图导航

知识目标

- 了解百度地图 API 的二次开发方法。

技能目标

- 实现地图显示、搜索地址、查询线路等功能。

【任务分析】

在互联网上有很多百度地图 API 有开发帮助学习文档，百度也有一个可供开发人员和学习者学习的百度地图开放平台，在这个平台上针对各种功能都有详细实例代码，学生可以根据需要的功能进行有针对性的学习，如图 8-5-1 所示。

网址是 http://developer.baidu.com/map/jsdemo.htm#a1_2。

图 8-5-1　百度地图开放平台

【任务实施】

① 加载百度地图。在 Map. jsp 添加代码如下。

```
<script
src="http://api.map.baidu.com/api?v=2.0&ak=R6jiNdN7wEGdC4bskfGYuKMbGrUDOfhO"
type="text/javascript"></script>
```

注意

相关秘钥可在百度地图平台注册获取。

② 通过 JavaScript 实现地图显示、搜索地址和查询线路的功能。百度地图在 Map. jsp 添加 JavaScript 代码如下。

```
var map = new BMap.Map("map");
var point = new BMap.Point(120.59646,31.226024);
    map.centerAndZoom(point, 15);
    map.enableScrollWheelZoom();
    function pos() {
        var geolocation = new BMap.Geolocation();
        geolocation.getCurrentPosition(function(r) {
            if(this.getStatus() == BMAP_STATUS_SUCCESS) {
                var mk = new BMap.Marker(r.point);
                map.addOverlay(mk);
                map.panTo(r.point);
            } else {
```

```
                alert('failed' + this.getStatus());
            }
        }, {enableHighAccuracy : true})
    }

    //添加带有定位的导航控件
    var navigationControl = new BMap.NavigationControl({
        // 靠左上角位置
        anchor : BMAP_ANCHOR_TOP_LEFT,
        // LARGE 类型
        type : BMAP_NAVIGATION_CONTROL_LARGE,
        // 启用显示定位
        enableGeolocation : true
    });
    map.addControl(navigationControl);

    //添加定位控件
    var geolocationControl = new BMap.GeolocationControl();
    geolocationControl.addEventListener("locationSuccess", function(e){
        // 定位成功事件
        var address = '';
        address += e.addressComponent.province;
        address += e.addressComponent.city;
        address += e.addressComponent.district;
        address += e.addressComponent.street;
        address += e.addressComponent.streetNumber;
        });
    geolocationControl.addEventListener("locationError", function(e){
        // 定位失败事件
        alert(e.message);
    });
    map.addControl(geolocationControl);

    function localsearch() {
        map.clearOverlays();
        $("#r-result").hide();
        $("#dh").hide();
        var loc = $("#localtion").val();
        var local = new BMap.LocalSearch(map, {
            renderOptions : {
                map : map,
                panel : "result"
```

```
            }
        });
        local.search(loc);
        $("#result").show();
        if (loc == "") {
            $("#result").hide();
            $("#dh").show();
        }
    }

    function clearsearch() {
        $("#localtion").val("");
        localsearch();
        $("#result").hide();
        $("#dh").show();
        $("#r-result").empty();
        $("#r-result").hide();
        $("#end").val("");
        $("#start").val("");
    }
var type = 0;
    function select() {
        $("#r-result").empty();
        var d= $("#drive").css("background-position-y");
        var b= $("#bus").css("background-position-y");
        var w= $("#walk").css("background-position-y");
        if(d=="-84px"){
            drive();
        }else if(b=="-51px"){
            bus();
        }else if(w=="-119px"){
            walk();
        }else{
            alert("请选择出行方式!");
            return false;
        }
        //var type = $("#choose").val();
        var start = $("#start").val();
        var end = $("#end").val();
        map.clearOverlays();
        if (start!= ""&& end!= "") {
            var routePolicy 1 = [ BMAP_DRIVING_POLICY_LEAST_TIME,
                    BMAP_DRIVING_POLICY_LEAST_DISTANCE,
```

```
BMAP_DRIVING_POLICY_AVOID_HIGHWAYS ];
            var driving = new BMap.DrivingRoute(map, {
            renderOptions : {map : map,panel : "r-result"},
            policy : 0
        });
        driving.clearResults();
        var routePolicy2 = [ BMAP_TRANSIT_POLICY_LEAST_TIME,
                             BMAP_TRANSIT_POLICY_LEAST_TRANSFER,
                             BMAP_TRANSIT_POLICY_LEAST_WALKING,
                             BMAP_TRANSIT_POLICY_AVOID_SUBWAYS ];
        var transit = new BMap.TransitRoute(map, {
                renderOptions : {
                map : map,
                panel : "r-result"
            },
        policy : 0
        });
        transit.clearResults();
        var walking = new BMap.WalkingRoute(map, {
            renderOptions : {map : map,panel : "r-result"},
            policy : 0
            });
        walking.clearResults();
        if (type == 1) {
            search(start, end, routePolicy1[0]);
            function search(start, end, route) {
                driving.setPolicy(route);
                driving.search(start, end);
            }
        } else if (type == 2) {
            search(start, end, routePolicy2[0]);
            function search(start, end, route) {
                transit.setPolicy(route);
                transit.search(start, end);
            }
        } else {
            walking.search(start, end);
        }
            $("#r-result").show();
    } else {
        alert("请输入起点、终点");
        $("#r-result").hide();
    }
```

```
        }

    function drive(){
        $("#drive").css("background-position","-0px -84px");
        $("#bus").css("background-position","-0px -34px");
        $("#walk").css("background-position","-0px -100px");
        type=1;
    }
    function bus(){
        $("#bus").css("background-position","-0px -51px");
        $("#drive").css("background-position","-0px -68px");
        $("#walk").css("background-position","-0px -100px");
        type=2;
    }
    function walk(){
        $("#walk").css("background-position","-0px -119px");
        $("#drive").css("background-position","-0px -68px");
        $("#bus").css("background-position","-0px -34px");
        type=3;
    }
```

技能训练

1. 制作首页车辆分布展示效果

任务描述

采用百度的 Echarts 图表组件，制作首页车辆分布地图，以中国地图为背景，实现新能源汽车在全国分布情况可视化展示。

任务分析

通过 Ajax 方法查询车辆在各省分布数据，作为 Echarts 图表的数据配置项。Echarts 地图配置项说明请参见 Echarts 官方网站。

2. 根据车牌号查询指定车辆详细信息

任务描述

根据车牌号查询指定车辆详细信息，页面效果如图 8-5-2 和图 8-5-3 所示。

图 8-5-2　指定车辆查询

任务分析

指定车辆查询，需根据用户输入的关键字进行检索，将车牌号作为查询的条件传入到控制层，控制层进行处理后将对应车牌的车辆信息在 CarInformation. jsp 页面中显示。

图 8-5-3　指定车辆详情

项目总结

在本项目中通过对车辆信息监控模块各功能的实现，重点阐述了以下几项任务技能。

- Spring+SpringMVC+MyBatis 框架的功能开发。
- 通过百度 Echarts 组件实现数据的可视化展示。
- 百度地图开发组件的使用。

本项目中的任务侧重于数据的图表化展示，不仅要通过框架技术的使用获取相关业务数据，还要求熟悉 Echarts 图表插件的使用，将数据以图形化的形式呈现，这要求学生有一定的自学能力并具备较熟练的前端开发 JQuery 技术，综合性较强，具有一定的难度。

项目9　车友圈模块

PPT 车友圈模块

项目描述

本项目将基于 Spring MVC+MyBatis 框架，实现车友圈模块的开发，具体功能包括添加车友、车友列表显示、车友圈信息发布和展示、点赞评论等功能。

知识目标

- 进一步熟悉 SpringMVC 框架的运行原理。
- 进一步熟悉 MyBatis 框架的运行原理。
- 进一步熟悉 JSP 页面通过 Ajax 技术请求及更新数据的方法。

技能目标

- 熟练运用 Bootstrap3 框架实现网页开发。
- 在 SpringMVC+MyBatis 开发框架上，结合前端开发技术 jQuery 进行熟练开发。

任务列表

任务编号	任务名称	建议课时
任务 9.1	添加车友	4
任务 9.2	显示车友列表	4
任务 9.3	发布车友圈信息	1
任务 9.4	展示车友圈信息	1
任务 9.5	实现车友圈点赞、评论功能	4
技能训练	删除车友功能 删除车友圈帖子和评论	1 1
	总计：	16 课时

任务 9.1　添加车友

【任务描述】

本任务通过 Ajax 技术和 SpringMVC 框架技术，实现车友圈模块的“添加车友”功能，该功能类似于添加微信好友，通过输入名称添加车友，并在“我的车友”列表中显示，车友不能重复添加，页面效果如图 9-1-1 所示。

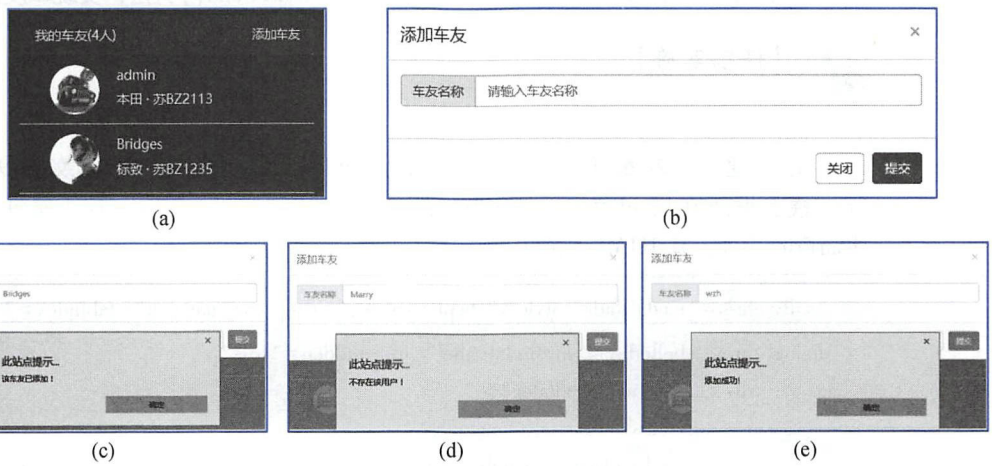


图 9-1-1　添加车友效果图

【任务目标】

知识目标

- 了解 Bootstrap 模态框的作用及用法。
- 进一步熟悉 POJO 类、对象持久化映射层类、业务逻辑处理层类的作用及创建方法。
- 进一步熟悉页面中 Ajax 异步请求的原理及实现方法。

技能目标

- POJO 类 CarFriend 的创建。
- 对象持久化映射层类 CarFriendMapper 的创建。
- 业务逻辑处理层类 CarFriendService 的创建。
- 控制层 MyRingController 中 addCarFriend( ) 方法的编写及请求调用。

微课 9-1
添加车友

【任务分析】

添加车友功能与项目 6 中的会员注册、项目 5 中的添加车辆的流程相似，首先判断该车友是否是系统合法用户，如不是则提示“不存在该用户!”，若用户合法，则进一步根据数据库车友表验证该用户是否已为车友，如是则提示“该车友已添

加!”，如不是，接收用户名参数信息，通过查询获取其对应的 ID，执行数据库添加操作，添加成功显示“添加成功!”。

具体的开发流程如下：

① 创建会员信息数据对应 POJO 类 User. java，控制层 UserController. java，业务逻辑处理层类 UserService. java，对象持久化映射层类 UserMapper. java、UserMapper. xml 等（项目 4 中已完成，本任务略去）。

② 创建车友信息数据对应 POJO 类 CarFriend. java，定义数据结构。

③ 创建对象持久化映射层类 CarFriendMapper. java、CarFriendMapper. xml。

④ 创建业务逻辑的处理层类 CarFriendService. Java，声明调用方法。

⑤ 编辑控制层 MyRingController. java，添加 addCarFriend( )方法，实现车友添加。

【任务实施】

(1) 创建“车友圈”页面 Myring. jsp

① 创建“车友圈页面” Myring. jsp，添加 Bootstrap 模态框，样式如图 9-1-1 所示。模态框所在层 id 为 car_manager，“提交”按钮的 onclick 事件属性值为 addCarFriend( )，其 HTML 代码如下。

```
<div class="modal fade" style="display:none;" id="car_manager" tabindex="-1" role
="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-dismiss="modal" aria-
hidden="true">&times;</button>
                <h4 class="modal-title" id="myModalLabel">添加车友</h4>
            </div>
            <div class="modal-body">
                <div class="input-group">
                    <span class="input-group-addon">车友名称</span>
                    <input type="text" id="carfname" class="form-control"
placeholder="请输入车友名称">
                </div>
                <br>
            </div>
            <div class="modal-footer">
        <button type="button" class="btn btn-default" data-dismiss="modal">关闭
</button>
        <button type="button" onclick="addCarFriend( )" class="btn btn-primary">提交
</button>
            </div>
        </div>
```

```
    </div>
</div>
```

② 在 Myring. jsp 页面相应位置添加“添加车友”超链接，代码如下：

```
<div class="container-fluid content">
    <div class="row">
        <div class="col-md-3 leftbox">
        <div class="list-group">
    <div class="list-group-item active" style="background-color:#196BA2;">
            <div class="pull-left" style="color:#F0F8FD">我的车友</div>
            <a class="pull-right" href="javascript:void(0)" data-toggle="modal"
            data-target="#car_manager" style="color:#F0F8FD">添加车友</a>
        </div>
        </div>
    </div>
</div>
```

(2) 创建车友信息数据对应 POJO 类 CarFriend. java

在 src 的 com. piesat. zyms. domain. core 包中创建类 CarFriend，类的属性如图 9-1-2 所示，自动生成 getters and setters。

```
public class CarFriend {

    private int id;
    private String carfname;
    private String cartype;
    private String carnumber;
    private String userid;
    private String friendid;
    private User user;
    private CarMessage carmessage;
```

图 9-1-2　CarFriend 类属性

(3) 实现对象持久化映射层 CarFriendMapper. java、CarFriendMapper. xml

① 在包 com. piesat. zyms. persistence 中新建接口 CarFriendMapper. java，定义添加车友方法 saveCarFriend，代码如下。

```
package com.piesat.zyms.persistence;
public interface CarFriendMapper {
    public void saveCarFriend(CarFriend carfriend);
}
```

② 继续在同一个包中新建文件 CarFriendMapper. xml，用于映射数据表 carfriend 的字段和表上的操作，代码如下。

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://my-
   batis.org/dtd/mybatis-3-mapper.dtd">
3. <mapper namespace="com.piesat.zyms.persistence.CarFriendMapper">
```

```
4.  <resultMap type="com.piesat.zyms.domain.core.CarFriend" id="carfriend">
5.      <id column="ID" property="id"/>
6.      <result column="CARFNAME" property="carfname"/>
7.      <result column="CARTYPE" property="cartype"/>
8.      <result column="CARNUMBER" property="carnumber"/>
9.      <result column="USERID" property="userid"/>
10.     <result column="FRIENDID" property="friendid"/>
11. <association property="user" javaType="User">
12.     <id column="ID" property="id"/>
13.     <result column="HPIC" property="hpic"/>
14.     <result column="USERNAME" property="username"/>
15. </association>
16. <association property="carmessage" javaType="CarMessage">
17.     <id column="id" property="id" />
18.     <result column="vehID" property="vehID" />
19.     <result column="plateNumber" property="plateNumber" />
20.     <result column="brand" property="brand" />
21.     <result column="remarks" property="remarks" />
22.     <result column="produce_date" property="produceDate" />
23.     <result column="current_mileage" property="currentMileage" />
24.     <result column="user_id" property="userId" />
25.     <result column="create_id" property="createId" />
26. </association>
27. </resultMap>
28.
29. <select id="getListByTId" parameterType="java.util.Map" resultMap="carfriend">
30.     select * from carfriend where userid=#{userid} and friendid = #{friendid}
31. </select>
32.
33. <insert id="saveCarFriend" parameterType="com.piesat.zyms.domain.core.CarFriend">
34. insert into carfriend (CARFNAME, CARTYPE, CARNUMBER, USERID, FRIENDID)
    values(#{carfname},#{cartype},#{carnumber},#{userid},#{friendid})
35. </insert>
36.
37. </mapper>
```

代码说明：

- 第 5 行~第 10 行：将数据表 carfriend 的字段与 POJO 类 CarFriend 的属性进行映射，因为表中的字段和类的属性名称不完全一致，此部分代码不可以省略。
- 第 11 行~第 15 行、第 16 行~第 26 行：association 标记表示在一个标签内部使用嵌套对象，关联的结果，根据查询的列和 resultMap 定义的对应关系来创建对象并写入值。

• 第 33 行~第 35 行：定义添加车友的方法 saveCarFriend( )，使用<insert></insert>标签，输入参数是 CarFriend，无返回值。

(4) 实现业务逻辑的处理层类 CarFriendService. java

在 src 文件夹包 com. piesat. zyms. service. cms 中新建类 CarFriendService，代码如下。

```
1. package com. piesat. zyms. service. cms;
2. @ Service
3. public class CarFriendService {
4. @ Autowired
5. private CarFriendMapper carfriendMapper;
6. public void saveCarFriend( CarFriend carfriend) {
7. carfriendMapper. saveCarFriend( carfriend) ;
8. }
9. public CarFriend getListByTId( Map<String,String> map) {
10.      return carfriendMapper. getListByTId( map) ;
11. }
12. }
```

(5) 实现视图层 Myring. jsp 的 Ajax 调用

在 Myring. jsp 页面中编写方法 addCarFriend( )，通过 Ajax 调用将用户信息传递给 Controller 并进行相应的处理，代码如下。

```
<div class="modal fade" style="display:none;" id="car_manager" tabindex="-1" role
="dialog" aria-labelledby="myModalLabel" aria-hidden="true">
      ......
      <div class="modal-footer">
      <button type="button" class="btn btn-default" data-dismiss="modal">关闭</button>
      <button type="button" onclick="addCarFriend()" class="btn btn-primary">提交</button>
      </div>
</div>
-------------------------------------------------------------------------------
<script type="text/javascript">
function addCarFriend(){
        var carfname = $("#carfname").val();
        $.ajax({
            type : "post",
            url : '<%=path %>'+"/ring/addcarfriend",
            dataType : "jsonp",
            jsonp:"jsoncallback",
            data : {
                carfname:carfname
            },
```

```
                success : function(data){
                    if(data==="error"){
                        alert("不存在该用户!");
                    }else if(data==="repeat"){
                        alert("该车友已添加!");
                    }else{
                        alert("添加成功!");
                        $("#car_manager").modal("hide");
                        $("#carfname").val("");
            window.location.href="http://" + location.host + "<%=path %>/" +"energy/myring";
                    }
                }
            });
        }
        </script>
```

(6) 实现控制层 MyRingController.java

在 com.piesat.zyms.web.cms 包中，新建 Java 类 MyRingController，代码如下。

```
package com.piesat.zyms.web.cms;
@RequestMapping("/ring")
@Controller
public class MyRingController {

    @Autowired
    private CarFriendService carfriendService;

    @Autowired
    private UserService userService;

    /**
     *车友圈添加车友
     */
    @RequestMapping("/addcarfriend")
    public void addCarFriend(HttpServletRequest req,HttpServletResponse resp)
        throws IOException,ServletException, ParseException {
            String carfname = req.getParameter("carfname");
            User user = userService.getUserByUsername(carfname);
            String result = "";
            if(user!=null){
                User us = (User)req.getSession().getAttribute("user");
                String userid = us.getId();
                String friendid = user.getId();
                String cartype="";
                String carnumber="";
```

```
                CarMessage cm=carMessageService.getCarMessageByUser(userid);
                if(cm!=null){
                    cartype=cm.getCarType();
                    carnumber=cm.getPlateNumber();
                }
                Map<String,String> map = new HashMap<String,String>();
                map.put("userid", userid);
                map.put("friendid", friendid);
                CarFriend caf = carfriendService.getListByTId(map);
                if(caf!=null){
                    result=new Gson().toJson("repeat");
                }else{
                    CarFriend carfriend = new CarFriend();
                    carfriend.setCarfname(carfname);
                    carfriend.setUserid(userid);
                    carfriend.setFriendid(friendid);
                    carfriend.setCartype(cartype);
                    carfriend.setCarnumber(carnumber);
                    carfriendService.saveCarFriend(carfriend);
                    result=new Gson().toJson(carfriend);
                }
            }else{
                result=new Gson().toJson("error");
            }
            String jsonp = req.getParameter("jsoncallback");
            resp.setCharacterEncoding("UTF-8");
            resp.setContentType("text/html");
            if(jsonp!= null){
                result = jsonp+"("+result+")";
                resp.getWriter().write(result);
            }else{
                resp.getWriter().write(result);
        }
    }
}
```

任务 9.2　显示车友列表

【任务描述】

“车友圈”页面左右区域显示当前登录用户的所有车友，当进入页面时，自动加载车友列表。当添加车友成功后，将即时刷新车友列表，本任务将实现车友列表的显示功能，页面效果如图 9-2-1 所示。

图 9-2-1　我的车友列表效果图

【任务目标】

知识目标

- 对象持久化映射层类、业务逻辑处理层类中方法的创建。
- 进一步熟悉页面中 Ajax 异步请求的原理及实现方法。

技能目标

- 对象持久化映射层类 CarFriendMapper 中的创建 getFriendByUserId( )、getCount( )方法。
- 业务逻辑处理层类 CarFriendService 的创建 getFriendByUserId( )、getCount( )方法。
- 控制层 MyRingController 中 myring( )方法的编写及请求调用。
- 视图层 Head. jsp 中 Ajax 方法 jumpCarFriend( )的创建及编写。
- 视图层 Myring. jsp 中车友列表的显示。

微课 9-2
显示车友列表

【任务分析】

- 在视图层 Head. jsp 中，编写 Ajax 方法 jumpCarFriend( ) 判断用户是否登录，并转发请求。
- 在 IndexController 中实现一个方法，获取当前用户的车友列表，并将请求转发至 car/Myring. jsp。
- 在 CarFriendService. java 中定义 getFriendByUserId、getCount 方法，获取车友列表及人数。
- 在 CarFriendMapper. java、CarFriendMapper. xml 中声明和定义用户表、车友表以及车辆信息表上的联合查询方法 getFriendByUserId 以及 getCount 方法。
- 在视图层 Myring. jsp 中，通过循环将车友列表显示在页面上。

【任务实施】

(1) 在视图层 Head. jsp 中，编写 Ajax 方法判断用户登录情况

在 Head. jsp 页面中，创建“车友圈”超链接，并编写 onclick 事件方法 jump-

CarFriend()，用于判断用户登录情况，若登录，则将请求转发至 http://localhost:8080/car/energy/myring 进一步处理，若未登录则跳转至登录页面。

“车友圈”超链接 HTML 代码如下。

```
<div class="navbar-header">
    <a class="navbar-brand" onclick="jumpCarFriend()" href="javascript:void(0);">车友圈</a>
</div>
```

① 定义 Ajax 方法 jumpCarFriend()，代码如下。

```
function jumpCarFriend(){
        $.ajax({
            type : "post",
            url : '<%=path %>' + "/energy/judgesession",
            data : {
            },
            dataType : "jsonp",//数据类型为 json
            jsonp:"jsoncallback",
            success : function(data){
                if(data=="success"){
                    window.location.href="http://" + location.host + "<%=path %>/"
+"energy/myring";
                }else{
                    window.location.href="http://" + location.host + "<%=path %>/"
+"energy/login";
                }
            }
        });
        }
```

② 通过 IndexController 中定义的方法 judgeSession 判断用户登录状况。

```
@RequestMapping(value = "/judgesession")
    public void judgeSession(HttpServletRequest req,HttpServletResponse resp)
            throws IOException,ServletException, ParseException {
        User us = (User)req.getSession().getAttribute("user");
        String result = "";
        if(us!= null){
            result=new Gson().toJson("success");
        }else{
            result=new Gson().toJson("fail");
        }
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
```

```
            resp. setContentType( "text/html" ) ;
            if( jsonp ! = null ) {
                result = jsonp+" ( "+result+" ) " ;
                resp. getWriter( ) . write( result ) ;
            } else {
                resp. getWriter( ) . write( result ) ;
            }
        }
```

(2) 在 IndexController 中定义 myring 方法

在 IndexController 中实现 myring 方法，用于获取车友列表，代码如下。

```
@ RequestMapping( "/energy" )
@ Controller
public class IndexController {
.....
@ RequestMapping( value = "/myring" , method = RequestMethod. GET)
    public ModelAndView myring( HttpServletRequest request,
        HttpServletResponse response, ModelMap model) {
        User us = ( User) request. getSession( ) . getAttribute( "user" ) ;
        String userid = us. getId( ) ;
        List<CarFriend> carfriends = carfriendService. getFriendByUserId( userid) ;
        for ( CarFriend f : carfriends) {
            User user2 = new User( ) ;
            String p = f. getUser( ) . getHpic( ) ;
            String n = f. getUser( ) . getUsername( ) ;
            if( p ! =null&&!p. equals( "" ) ) {
                String hpic = "upload" +
f. getUser( ) . getHpic( ) . substring( f. getUser( ) . getHpic( ) . lastIndexOf( '\\' ) ) ;
                user2. setHpic( hpic) ;
                user2. setUsername( n) ;
                f. setUser( user2) ;
            }
        }
        int count = carfriendService. getCount( userid) ;
        ModelAndView mv = new ModelAndView( "car/Myring. jsp" ," carfriends" ,
carfriends) ;
        mv. addObject( "count" ,count) ;
        return mv;
    }
}
```

(3) 实现业务逻辑的处理层类 CarFriendService. java

在 CarFriendService. java 中添加方法 getFriendByUserId(Stringuserid)，方法 getCount(String userid)，代码如下。

```
package com. piesat. zyms. service. cms;
@ Service
public class CarFriendService {
    @ Autowired
    private CarFriendMapper carfriendMapper;
    ……
    public List<CarFriend> getFriendByUserId( String userid) {
        return carfriendMapper. getFriendByUserId( userid) ;
    }
public int getCount( String userid) {
        return carfriendMapper. getCount( userid) ;
    }
}
```

(4) 实现对象持久化映射层 CarFriendMapper. java、CarFriendMapper. xml

① 在 CarFriendMapper. java 中，声明获取车友列表、车友人数的方法，代码如下。

```
package com. piesat. zyms. persistence;
public interface CarFriendMapper {
  public List<CarFriend>getFriendByUserId ( String userid) ;
  public intgetCount ( String userid) ;
}
```

② 在 CarFriendMapper. xml 文件中，增加获取车友列表和人数的方法定义，代码如下。

```
<select id = " getFriendByUserId" parameterType = " java. lang. String" resultMap = " carfriend" >
        select * from carfriend f LEFT JOIN user u on u. id = f. friendid
        LEFT JOIN carmessage c on c. userid = f. friendid where f. userid = #{userid}
</select>
<select id = " getCount" resultType = " java. lang. Integer" >
        select count( * ) from carfriend where userid = #{userid}
</select>
```

(5) 视图层 Myring. jsp 中显示车友列表

在 Myring. jsp 页面中，读取控制层方法返回的 carfriends 及 count 对象，通过循环在页面输出车友列表，代码如下。

```
<divclass = " list-group-item active" style = " background-color:#196BA2;" >
        <div class = " pull-left" style = " color:#F0F8FD" >我的车友 (${count}人) </div>
        <a class = " pull-right" href = " javascript:void(0)" data-toggle = " modal"
data-target = " #car_manager" style = " color:#F0F8FD" >添加车友</a>
            <div class = " clearfix" ></div>
        </div>
    <div id = " carlist" >
```

```
<c:forEach items="${carfriends}" var="var" varStatus="vs">
<div class="list-group-item list-group-item-content">
    <div class="col-md-3">
        <img src="${var.user.hpic}" class="img-circle" border="0"
height="50" id="pic" width="50"/>
    </div>
    <div class="col-md-9">
    <div>${var.user.username}</div>
    <div style="margin-top:5px;">
        <span style="float:left;">${var.carmessage.brand}</span>
        <span style="float:left;">${var.carmessage.plateNumber}</span>
    </div>
    </div>
    <div class="clearfix"></div>
</div>
</c:forEach>
</div>
```

任务 9.3　发布车友圈信息

【任务描述】

本任务将实现车友圈模块的“发布信息”功能，合法用户通过在线编辑器发表带格式的文本，发布成功后会在“车友分享”列表中显示，页面效果如图 9-3-1 和图 9-3-2 所示。

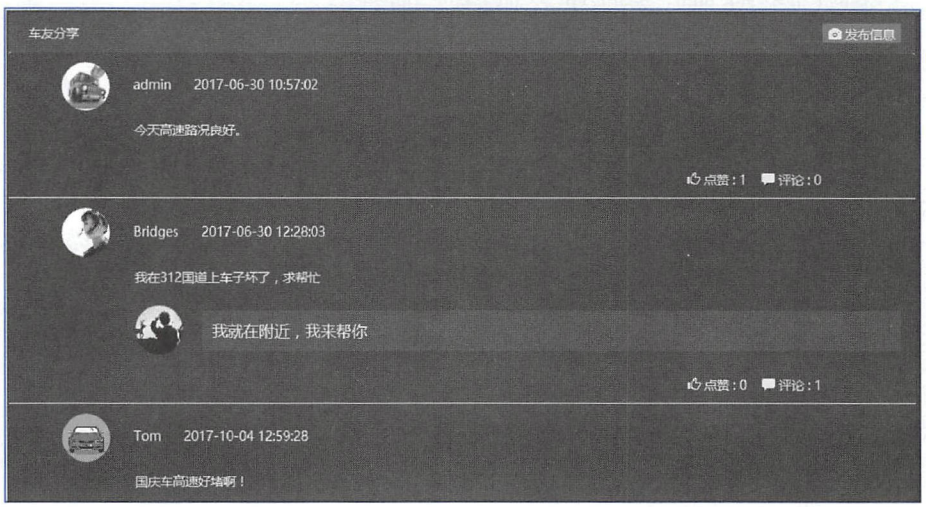


图 9-3-1　车友分享列表

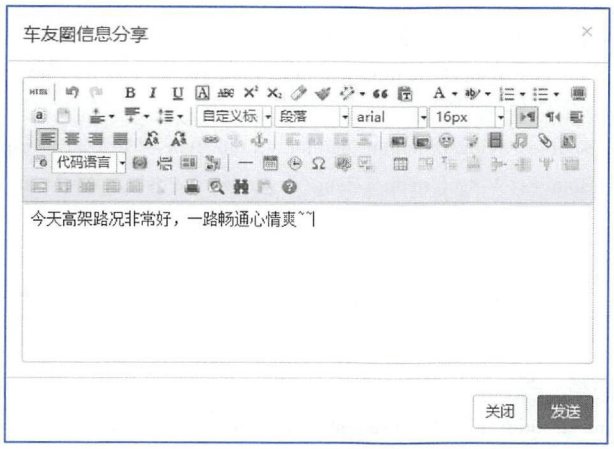


图 9-3-2　车友圈发布信息

微课 9-3
发布车友圈信息

【任务目标】

知识目标

- 了解 Bootstrap 模态框的使用方法。
- 了解 HTML 编辑器 ueditor。
- 进一步熟悉控制层、对象持久化映射层类、业务逻辑处理层类中的方法的实现。
- 进一步熟悉页面中 Ajax 异步请求的原理及实现方法。

技能目标

- 初步掌握文本编辑器 ueditor 的使用方法。
- 掌握 Spring MVC+MyBatis 框架下数据的保存。

【任务分析】

- 视图层 Myring. jsp 页面设计，单击对话框中“发送”按钮触发信息保存方法 saveContent( )。
- 创建车友圈信息数据对应 POJO 类 Content. java，定义数据结构。
- 在 MyRingController 中实现 saveContent( ) 方法，用于信息的保存。
- 创建业务逻辑的处理层类 contentService. java，同样实现一个 saveContent( ) 同名方法。
- 在 contentMapper. java、contentMapper. xml 中分别声明和定义车友圈信息表上的操作方法 saveContent 方法，实现信息的发布。

【任务实施】

(1) 在视图层 Myring. jsp 页面编辑

编辑“车友圈”页面 Myring. jsp，添加“信息发布”模态框，单击“发布信息”按钮弹出模态框，此模态框包含一个如图 9-3-2 所示的在线 HTML 编辑器 ueditor，代码如下。

```
<!--文本编辑器 ueditor 文件引用,文件可在官网下载 -->
<script src="resources/Widget/ueditor/ueditor.config.js" type="text/javascript" >
</script>
<script src="resources/Widget/ueditor/ueditor.all.js" type="text/javascript" ></script>
<script src="resources/Widget/ueditor/lang/zh-cn/zh-cn.js" type="text/javascript" >
</script>
<!--发布信息按钮-->
<button type="button" data-toggle="modal" data-target="#carContent" >发布信息
</button>
<!--模态对话框-->
<div class="modal fade" id="carContent " tabindex="-1" role="dialog" aria-labelledby
="myModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            ......
            <div class="modal-body">
                <script id="container" type="text/plain"></script>
            </div>
            <div class="modal-footer">
                <button type="button" class="btn btn-default" data-dismiss=
"modal">关闭
                </button>
                <button type="button" onclick ="saveContent()" class="btn btn-pri-
mary">发送
                </button>
            </div>
        </div>
    </div>
</div>
```

HTML 编辑器 ueditor 的配置如下。

```
<script type="text/javascript">
var editor;
$(function() {
    UE.Editor.prototype._bkGetActionUrl=UE.Editor.prototype.getActionUrl;
    UE.Editor.prototype.getActionUrl=function(action){
        if (action == 'uploadimage'){
            return "<%=path%>"+"/ueditor/upload.do?Type=Image&mark=1";
        }else{
            return this._bkGetActionUrl.call(this, action);
        }
```

```
        };
        editor = UE.getEditor('container',{
    toolbars:[['bold','indent','italic','underline','fontborder','simpleupload','justifyleft',
'justifyright','justifycenter','justifyjustify','forecolor','backcolor']],
                    wordCount:false,
                    elementPathEnabled:false,
                });
    });
    </script>
```

（2）编写 Ajax 方法 saveContent()，该方法由“提交”按钮触发，将转发请求至控制层，方法代码如下。

```
    <script type="text/javascript">
        function saveContent(){
            var txt = editor.getContent();
            var text= encodeURIComponent(txt);
            $.ajax({
                type : "post",
                url : '<%=path %>'+"/ring/saveContent",
                data : {
                    txt : text,
                },
                success : function(data){
                    alert("发送成功!")
                    window.location.href = "http://" + location.host + "<%=path %>/"
+ data;
                }
            });
        }
    </script>
```

（3）创建车友圈信息数据对应 POJO 类 Content.java

在 src 的 com.piesat.zyms.domain.core 包中创建类 Content，类的属性如图 9-3-3 所示，生成 getters and setters。

```
public class Content {

    private int id;
    private String txt;      //内容
    private int praise;      //是否点赞
    private int comments;    //评论数
    private Date releasetime; //发布时间
    private String userid;   //用户编号
    private List<Comment> comment; //评论列表
    private User user; //用户
```

图 9-3-3　Content 类属性

（4）在控制层 MyRingController 中实现 saveContent()方法

```
package com. piesat. zyms. web. cms;
@ RequestMapping( "/ring" )
@ Controller
public class MyRingController {
……
@ Autowired
private ContentService contentService;
    /* 保存车友圈发布信息 */
    @ RequestMapping( "/saveContent" )
     public void saveContent ( HttpServletRequest req, HttpServletResponse resp, ModelMap
model,String txt) throws IOException,ServletException, ParseException {
          String result = java. net. URLDecoder. decode( txt," UTF-8" ) ;
          Content content = new Content( ) ;
          content. setTxt( result) ;
          content. setReleasetime( new Date( ) ) ;
          User user = ( User) req. getSession( ) . getAttribute( " user" ) ;
          String userid = user. getId( ) ;
          content. setUserid( userid) ;
          contentService. saveContent( content) ;
          resp. setCharacterEncoding( " UTF-8" ) ;
          resp. getWriter( ) . write( " energy/myring" ) ;
          resp. getWriter( ) . flush( ) ;
          resp. getWriter( ) . close( ) ;
    }
}
```

(5) 创建业务逻辑的处理层类 ContentService. java

在 src 文件夹包 com. piesat. zyms. service. cms 中新建类 ContentService，代码如下。

```
package com. piesat. zyms. service. cms;
@ Service
public class ContentService {
@ Autowired
private ContentMapper contentMapper;
public void saveContent ( Content content) {
    contentMapper. saveContent( content) ;
}
}
```

(6) 实现对象持久化映射层 ContentMapper. java、ContentMapper. xml

① 在包 com. piesat. zyms. persistence 中新建接口 ContentMapper. java，定义添加车友圈信息方法 saveContent，代码如下。

```
package com. piesat. zyms. persistence;
public interface ContentMapper {
```

```
    public void saveContent(Content content);
}
```

② 继续在同一个包中新建文件 ContentMapper.xml，用于映射数据表 content 的字段和表上的操作，代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.piesat.zyms.persistence.ContentMapper">
    <resultMap type="com.piesat.zyms.domain.core.Content" id="content">
        <id column="ID" property="id"/>
        <result column="TXT" property="txt"/>
        <result column="PRAISE" property="praise"/>
        <result column="COMMENTS" property="comments"/>
        <result column="RELEASETIME" property="releasetime"/>
        <result column="CARFRIENDID" property="carfriendid"/>
        <result column="USERID" property="userid"/>
    <association property="user" column="userid" javaType="User">
        <id column="ID" property="id"/>
        <result column="HPIC" property="hpic"/>
        <result column="USERNAME" property="username"/>
    </association>
    <collection property="comment" ofType="Comment">
        <id column="MID" property="mid"/>
        <result column="TEXT" property="text"/>
        <result column="CONTENTID" property="contentid"/>
        <result column="TIME" property="time"/>
    </collection>
    </resultMap>
    <insert id="saveContent" parameterType="com.piesat.zyms.domain.core.Content">
        insert into content(TXT,PRAISE,COMMENTS,RELEASETIME,USERID) values(#{txt},#{praise},#{comments},#{releasetime},#{userid})
    </insert>
</mapper>
```

任务 9.4　展示车友圈信息

【任务描述】

本任务将实现车友圈信息的列表显示功能，列表中包含发表用户基本信息，车友圈信息（下称帖子）、车友回复、点赞数、评论数等，如图 9-4-1 所示。

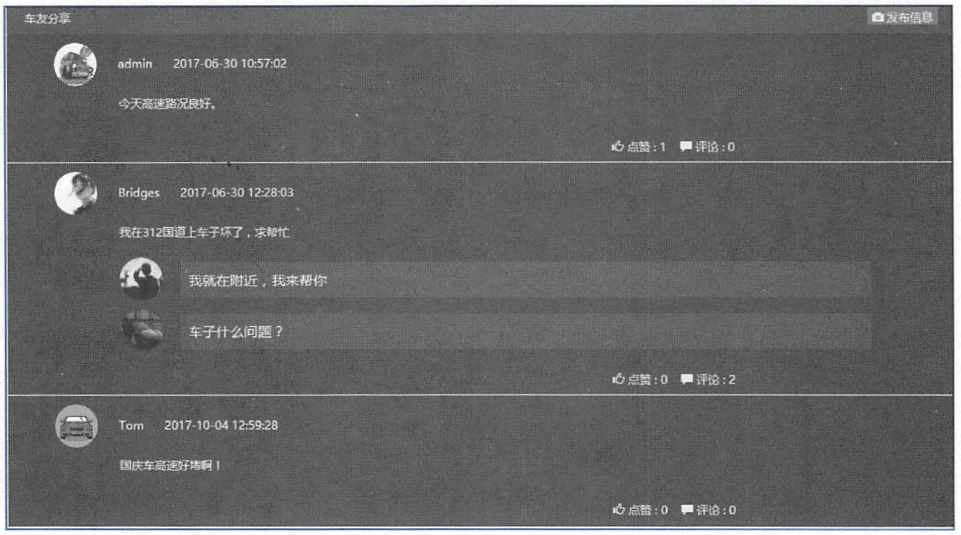


图 9-4-1　车友圈信息的列表显示

【任务目标】

知识目标

- 进一步熟悉控制层、对象持久化映射层类、业务逻辑处理层类中的方法的实现；
- 进一步熟悉在 JSP 页面中 Ajax 异步请求数据的原理及实现方法。

技能目标

- Spring MVC+MyBatis 框架下数据的读取操作。
- JSP 页面中车友圈信息数据的列表显示。

【任务分析】

微课 9-4
展示车友圈信息

- 创建评论信息数据对应 POJO 类 Comment. java。
- 在 IndexController 中的 myring 方法中，获取所有帖子信息及评论信息。
- 编辑业务逻辑的处理层类 ContentService. java、CommentService. java。
- 实现对象持久化映射层 ContentMapper. java、ContentMapper. xml 以及 CommentMapper. java 和 CommentMapper. xml。
- 在视图层 Myring. jsp 中，通过循环方式将方法返回的帖子及回复列表显示在页面上。

【任务实施】

① 创建评论信息数据对应 POJO 类 Comment. java。

在 com. piesat. zyms. domain. core 包中创建类 Comment，类的属性如图 9-4-2 所示，自动生成 getters and setters。

② 在 IndexController 中编辑 myring 方法。

```
public class Comment {

    private int id;
    private String text;
    private int contentid;
    private Date time;
    private String userid;
    private User user;
```

图 9-4-2　Comment 类

在 IndexController 类已实现 myring 方法中，添加如下代码，获取所有帖子及评论信息。

```
public ModelAndView myring(HttpServletRequest request,
    HttpServletResponse response, ModelMap model) {
    ……
    //获取所有帖子列表
    List<Content> contents = contentService.getContentByfid(userid);
    for (Content o : contents) {
        User user1 = new User();
        String p = o.getUser().getHpic();
        String n = o.getUser().getUsername();
        if(p!=null&&!p.equals("")){
            String hpic = "upload" +
o.getUser().getHpic().substring(o.getUser().getHpic().lastIndexOf('\\'));
            user1.setHpic(hpic);
            user1.setUsername(n);
            o.setUser(user1);
        }
    }
    //获取所有评论列表
    List<Comment> comments = commentService.getComment();
    for (Comment s : comments) {
        User user3 = new User();
        String p = s.getUser().getHpic();
        String n = s.getUser().getUsername();
        if(p!=null&&!p.equals("")){
            String hpic = "upload"
+s.getUser().getHpic().substring(s.getUser().getHpic().lastIndexOf('\\'));
            user3.setHpic(hpic);
            user3.setUsername(n);
            s.setUser(user3);
        }
    }
    ModelAndView mv = new ModelAndView("car/Myring.jsp","carfriends",carfriends);
            mv.addObject("comments", comments);
            mv.addObject("contents",contents);
            mv.addObject("count",count);
            return mv;
}
```

③ 编辑业务逻辑的处理层类 ContentService.java、CommentService.java。

在 ContentService.java 中添加方法 getContentByfid(String userid)，获取车友圈帖子，返回一个列表，代码如下。

```
package com.piesat.zyms.service.cms;
@Service
public class ContentService {
    @Autowired
    private ContentMappercontentMapper;
        ……
    public List<Content> getContentByfid(String userid){
        return contentMapper.getContentByfid(userid);
    }
}
```

创建 CommentService.java，添加方法 getComment()，获取帖子评论，同样返回一个列表，代码如下。

```
@Service
public class CommentService{
    @Autowired
    private CommentMapper commentMapper;
    ……
    public List<Comment> getComment(){
        return commentMapper.getComment();
    }
}
```

④ 实现对象持久化映射层 ContentMapper.java、ContentMapper.xml 以及 CommentMapper.java 和 CommentMapper.xml。

在 ContentMapper.java 和 CommentMapper.java 中，分别声明获取帖子和评论的方法，代码如下。

```
package com.piesat.zyms.persistence;
public interface ContentMapper {
    public List<Content>getContentByfid(String userid);
}
public interface CommentMapper {
    public List<Comment>getComment();
}
```

在 ContentMapper.xml 文件中，增加获取帖子列表方法的定义，代码如下。

```
<select id="getContentByfid" parameterType="java.lang.String" resultMap="content">
        (select c.*,u.hpic,u.username from content c LEFT JOIN user u on c.userid
= u.id
        where c.userid in (select f.friendid from carfriend f where f.userid = #{userid})
order by releasetime desc)
        union
```

```
        (select c. *, u. hpic, u. username from content c LEFT JOIN user u on c. userid
= u. id
        where c. userid in (#{userid})  order by releasetime desc)
    </select>
```

CommentMapper. xml 文件需要创建，然后添加获取评论列表的方法的定义，完整代码如下。

```
<?xml version="1. 0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis. org//DTD Mapper 3. 0//EN" "http://myba-
tis. org/dtd/mybatis-3-mapper. dtd">
<mapper namespace="com. piesat. zyms. persistence. CommentMapper">
    <resultMap type="com. piesat. zyms. domain. core. Comment" id="comment">
        <id column="ID" property="id"/>
        <result column="TEXT" property="text"/>
        <result column="CONTENTID" property="contentid"/>
        <result column="TIME" property="time"/>
        <result column="USERID" property="userid"/>
    <association property="user" javaType="User">
        <id column="ID" property="id"/>
        <result column="HPIC" property="hpic"/>
        <result column="USERNAME" property="username"/>
    </association>
    </resultMap>

    <select id="getComment" resultMap="comment">
        select * from comment c LEFT JOIN user u on u. id=c. userid
    </select>
</mapper>
```

⑤ 视图层 Myring. jsp 中显示帖子及评论列表。

在 Myring. jsp 页面中，读取方法返回的 contents 和 comments 对象，使用 JSTL 的迭代标签在页面输出，代码如下。

```
<div id="show" class="list-group">
    <c:forEach items="${contents}" var="var" varStatus="vs">
        <div id="divs${vs. count}" class="list-group-item">
        <input id="ids${vs. count}" value="${var. id}" type="hidden" />
        <div style="margin-left:40px;">
            <img src="${var. user. hpic }" class="img-circle" id="pic"/>
            <span style="padding-left: 20px">${var. user. username}</span>
            <span style="margin-left:20px;">
<fmt:formatDate value="${var. releasetime}" pattern="yyyy-MM-dd HH:mm:ss"/>
```

```
                    </span>
                </div>
                    <div style="margin-left:115px;margin-top:10px;">${var.txt}</div>
                        <div style="margin-left:115px;margin-top:10px;">
                        <table>
                        <tbody id="co ${vs.count}">
                        <c:forEach items="${comments}" var="car" varStatus="cs">
                            <c:if test="${car.contentid==var.id}">
                                <tr>
                                <td>
                                <spanstyle="float:left;width:60px;margin-top:10px;">
    <img src="${car.user.hpic }" class="img-circle" border="0" height="50" id="pic"
/></span>
                                <span>${car.text}</span>
                                </td>
                                </tr>
                            </c:if>
                        </c :forEach>
                        </tbody>
                        </table>
                        </div>
                    <div style="margin-top:20px;margin-left:680px;">
                        <span class="glyphicon glyphicon-thumbs-up"></span>点赞:
                        <span>${var.praise}</span>
                        <span class="glyphicon glyphicon-comment"></span>评论:
                        <span>${var.comments}</span>
                        </div>
                    </div>
            </c:forEach>
        </div>
```

任务 9.5　实现车友圈点赞、评论功能

【任务描述】

本任务将实现车友圈的点赞和评论功能，用户可以对帖子进行点赞和回复评论，评论不限次数，但不能重复点赞，页面效果如图 9-5-1 所示。

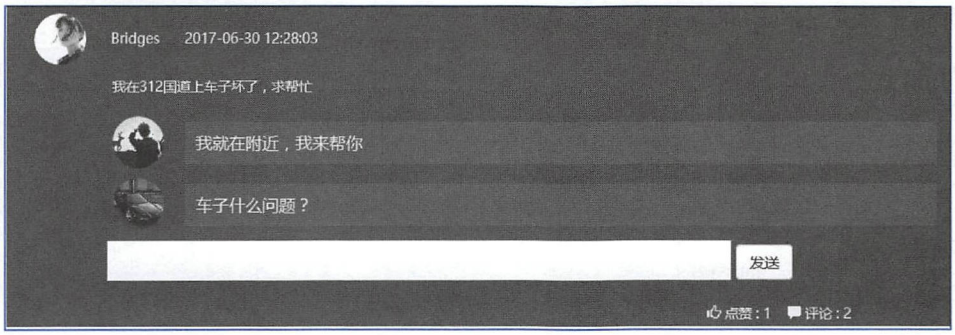


图 9-5-1　点赞和发表评论效果图

【任务目标】

微课 9-5
车友圈点赞、评论功能

知识目标

- 熟悉控制层、对象持久化映射层类、业务逻辑处理层类中的方法的实现。
- 熟悉页面中 Ajax 异步请求的原理及实现方法。

技能目标

- 掌握 Spring MVC+MyBatis 框架下数据的更新和保存。

【任务分析】

- 在 MyRingController 中创建 addpraise、sendcomment、addcomment 方法，分别实现点赞累加、添加评论内容和评论数累加。
- 编辑业务逻辑的处理层类 ContentService. java、CommandService. java。
- 实现对象持久化映射层 ContentMapper. java、ContentMapper. xml 以及 CommentMapper. java 和 CommentMapper. xml。

【任务实施】

① 在视图层 Myring. jsp 页面添加“点赞”和“评论”按钮，HTML 代码如下。

```
<button id="praise${vs.count}" onclick="praise(${vs.count})" type="button" >
        <span class="glyphicon glyphicon-thumbs-up"></span>点赞 :
        <span> ${var.praise}</span>
</button>
<button id="comment${vs.count}" onclick="comments(${vs.count})" type=
"button" >
        <span class="glyphicon glyphicon-comment"></span>评论 :
        <span> ${var.comments}</span>
</button>
```

以上代码中，“点赞”按钮触发 praise( )方法，“评论”按钮触发 comments( )方法。两个方法的参数 count 都用于标识被点赞或评论的帖子所在层的 id。praise 方法

将请求提交给 ring/addpraise，js 代码如下：

```
function praise(count){
    var pid = $("#ids"+count).val();
    $.ajax({
        type : "post",
        url : '<%=path %>' + "/ring/addpraise",
        data : {
            pid : pid
        },
        dataType : "jsonp",//数据类型为 json
        jsonp:"jsoncallback",
        success : function(data){
            if(data==="repeat"){
                alert("该用户已点过赞!");
            }else{
                var str = "";
                str+="<span class=\"glyphicon glyphicon-thumbs-up\"></span>
点赞 : <span>"+data.praise+"</span>";
                $("#praise"+count).html(str);
            }
        }
    });
}
```

comments()方法比较简单，将显示原本隐藏的层 din，用于编辑和发送评论，方法代码如下。

```
function comments(count){
    $("#din"+count).show();
}
```

din 层外观如下图 9-5-2 所示，HTML 代码如下。

```
<div id="din${vs.count}" style="display:none;margin-top:10px;margin-left:110px;">
    <input id="inputs${vs.count}" style="color:black;width:600px;height:40px;" />
    <button onclick="send(${vs.count})" type="button" class="btn btn-default">发
送</button>
</div>
```

图 9-5-2　din 层外观

“发送”按钮将触发 send 方法，将请求发送至 ring/sendcomment，对车友发表的评论进行保存，保存成功后，隐藏 din 层，并通过 addcomments 方法更新帖子评论总

数。send 方法的代码如下。

```
function send(count){
    var cid = $("#ids"+count).val();
    var text = $("#inputs"+count).val();
    $.ajax({
        type : "post",
        url : '<%=path %>' + "/ring/sendcomment",
        data : {
            cid : cid,
            text : text
        },
        dataType : "jsonp",//数据类型为 json
        jsonp:"jsoncallback",
        success : function(data){
        $("#inputs"+count).val("");
        $("#din"+count).hide();
        addcomments(count);
        var str = "";
        str+="<tr>";
        str+="<td>";
        str+="<span style=\"float:left;width:60px;margin-top:10px;\"><img src="+
data.hpic+" class=\"img-circle\" border=\"0\" height=\"50\" id=\"pic\" width=\"50\"
/></span>";
        str+="<span style=\"display:inline-block;width:800px;border:0px solid;
padding:10px;margin-left:10px;margin-top:15px;background-color:#196BA2;color:#F0F8FD;
\">"+data.text+"</span>";
        str+="</td>";
        str+="</tr>";
        $("#co"+count).append(str);
    }
    });
}
```

addcomments 方法代码如下：

```
function addcomments(count){
    var cid = $("#ids"+count).val();
    $.ajax({
        type : "post",
        url : '<%=path %>' + "/ring/addcomment",
        data : {
            cid : cid
        },
```

```
                dataType : "jsonp",//数据类型为 json
                jsonp:"jsoncallback",
                success : function(data){
                    var str = "";
                    str+="<span class=\"glyphicon glyphicon-comment\"></span>评论 :
<span>"+data.comments+"</span>";
                $("#comment"+count).html(str);
            }
        });
        }
```

② 在 MyRingController 中添加处理请求的方法。

在控制层 MyRingController 中添加 addpraise 方法，实现点赞数的累加。该方法还需要判断用户是否已对帖子进行过点赞，代码如下。

```
@RequestMapping("/addpraise")
public void addpraise(HttpServletRequest req,HttpServletResponse resp)
    throws IOException,ServletException, ParseException {
        int pid = Integer.parseInt(req.getParameter("pid"));
        Content praise = contentService.getContentById(pid);
        User user = (User)req.getSession().getAttribute("user");
        String userid = user.getId();
        Map<String,Object> map = new HashMap<String,Object>();
        map.put("contentid", pid);
        map.put("userid", userid);
        ContentPraise cp = contentpraiseService.getContentPraiseByUserId(map);
        String result = "";
        if(cp!=null){
            result=new Gson().toJson("repeat");
        }else{
            int count = praise.getPraise();
            count++;
            praise.setPraise(count);
            ContentPraise contentpraise = new ContentPraise();
            contentpraise.setContentid(pid);
            contentpraise.setUserid(userid);
            contentpraiseService.saveContentPraise(contentpraise);
            contentService.updateContent(praise);
            result=new Gson().toJson(praise);
        }
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp!= null){
```

```
            result = jsonp+"("+result+")";
            resp.getWriter().write(result);
        }else{
                resp.getWriter().write(result);
        }
    }
```

方法中的 ContentPraise 类比较简单，包含 id、contentid、userid 这 3 个属性，此处不再赘述。

接下来接续在 MyRingController 中添加评论相关方法 sendcomment 和 addcomment，前者用来保存评论的信息，后者的作用是更新评论数量，实现代码如下。

sendcomment 方法：

```
@RequestMapping("/sendcomment")
public void sendcomment(HttpServletRequest req,HttpServletResponse resp)
    throws IOException,ServletException, ParseException {
        int cid = Integer.parseInt(req.getParameter("cid"));
        String text = req.getParameter("text");
        User user = (User)req.getSession().getAttribute("user");
        String userid = user.getId();
        Comment comment = new Comment();
        comment.setText(text);
        comment.setContentid(cid);
        comment.setTime(new Date());
        comment.setUserid(userid);
        User u = userService.getUserById(userid);
        String h = u.getHpic();
        String hpic = "\\"+"upload" + h.substring(h.lastIndexOf('\\'));
        commentService.saveComment(comment);
        Map<String,Object> map = new HashMap<String,Object>();
        map.put("hpic", hpic);
        map.put("text",text);
        String result=new Gson().toJson(map);
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp!= null){
            result = jsonp+"("+result+")";
            resp.getWriter().write(result);
        }else{
            resp.getWriter().write(result);
        }
    }
```

addcomment 方法：

```
@RequestMapping("/addcomment")
public void addcomment(HttpServletRequest req,HttpServletResponse resp)
    throws IOException,ServletException, ParseException {
        int cid = Integer.parseInt(req.getParameter("cid"));
        Content comment = contentService.getContentById(cid);
        int count = comment.getComments();
        count++;
        comment.setComments(count);
        contentService.updateContent(comment);
        String result=new Gson().toJson(comment);
        String jsonp = req.getParameter("jsoncallback");
        resp.setCharacterEncoding("UTF-8");
        resp.setContentType("text/html");
        if(jsonp!= null){
            result = jsonp+"("+result+")";
            resp.getWriter().write(result);
        }else{
            resp.getWriter().write(result);
        }
    }
```

③ 编写业务逻辑的处理层类。

在已有的 ContentService.java 中添加方法 getContentById(int id)，根据帖子编号返回被点赞或评论的 Content 对象，添加方法 updateContent(Content content)更新帖子，代码如下。

```
package com.piesat.zyms.service.cms;
@Service
public class ContentService {
    @Autowired
    private ContentMapper contentMapper;
        ……
    public Content getContentById(int id){
        return contentMapper.getContentById(id);
    }
    public void updateContent(Content content){
        contentMapper.updateContent(content);
    }
}
```

创建业务逻辑的处理层类 ContentPraiseService.java，添加方法 saveContentPraise()，用来记录为帖子点赞的用户 ID，代码如下。

```
package com. piesat. zyms. service. cms;
@ Service
public class ContentPraiseService {
    @ Autowired
    private ContentPraiseMapper contentpraiseMapper;
        ……
    public void saveContentPraise( ContentPraise contentpraise) {
        contentpraiseMapper. saveContentPraise( contentpraise) ;
    }
}
```

在 CommentService. java 中添加方法 saveComment( )，保存评论，代码如下。

```
@ Service
public class CommentService{
    @ Autowired
    private CommentMapper commentMapper;
    ……
    public void saveComment( Comment comment) {
        commentMapper. saveComment( comment) ;
    }
}
```

④ 实现对象持久化映射层。

在已有 ContentMapper. java 中，声明 getContentById 方法、updateContent 方法，代码如下。

```
package com. piesat. zyms. persistence;
public interface ContentMapper {
    ……
    public Content getContentById ( int id) ;
    public void updateContent ( Content content) ;
}
```

在 CommentMapper. java 中，声明 saveComment 方法，代码如下：

```
public interface CommentMapper {
    public void saveComment ( Comment comment) ;
}
```

创建 ContentPraiseMapper 接口，声明方法 saveContentPraise，代码如下。

```
public interface ContentPraiseMapper {
    public void saveContentPraise( ContentPraise contentpraise) ;
}
```

在 ContentMapper. xml 文件中，增加根据编号获取帖子方法及更新帖子基本信息方法的定义，代码如下。

```
<select id="getContentById" parameterType="java.lang.Integer" resultMap="content">
        select * from content where id = #{id} order by releasetime desc
</select>

<update id="updateContent" parameterType="com.piesat.zyms.domain.core.Content">
    update content set
    TXT=#{txt},PRAISE=#{praise},COMMENTS=#{comments},RELEASETIME=#{releasetime} where id=#{id}
</update>
```

在 CommentMapper.xml 文件中，添加保存评论的方法定义，代码如下。

```
<insert id="saveComment" parameterType="com.piesat.zyms.domain.core.Comment">
        insert into comment(TEXT,CONTENTID,TIME,USERID)
    values(#{text},#{contentid},#{time},#{userid})
</insert>
```

创建 ContentPraiseMapper.xml 文件，添加保存用户对帖子点赞情况的方法定义，完整代码如下。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.piesat.zyms.persistence.ContentPraiseMapper">
    <resultMap type="com.piesat.zyms.domain.core.ContentPraise" id="contentpraise">
        <id column="ID" property="id"/>
        <result column="CONTENTID" property="contentid"/>
        <result column="USERID" property="userid"/>
    </resultMap>

    <insert id="saveContentPraise"
parameterType="com.piesat.zyms.domain.core.ContentPraise">
        insert into contentpraise(CONTENTID,USERID) values(#{contentid},#{userid})
    </insert>
</mapper>
```

技能训练

1. 删除车友功能

任务描述

在每一个车友列表项中创建“删除”按钮，实现删除车友功能。

任务分析

编写 jQuery 方法 DeleteCarFriend()，通过 Ajax 方法提交至 Controller 实现车友的删除，在“删除”按钮的 Onclick 事件中调用 DeleteCarFriend()方法。

2. 删除车友圈帖子和评论

任务描述

在车友圈信息列表和评论列表中创建“删除”按钮，实现删除车友圈信息和删除评论功能，登录用户只能删除自己发布的车友圈信息和评论。

任务分析

编写 jQuery 方法 DeleteMessage( ) 和 DeleteCommon( )，通过 Ajax 方法提交至 Controller 实现车友的删除，在“删除”按钮的 Onclick 事件中调用 DeleteMessage( ) 和 DeleteCommon( ) 方法。

项目总结

本项目通过 Spring MVC+MyBatis 框架技术介绍了车友圈模块的几个主要功能的实现方法，包括添加车友、车友列表显示、车友圈信息发布、展示、点赞及评论等功能，在前几个项目学习的基础上，本项目可进一步巩固和提高学习者对于 Spring MVC+MyBatis 框架开发的熟练程度，增强 Web 开发的综合应用能力。

第3篇 云平台部署发布篇

项目10 云平台部署发布

PPT 云平台部署发布

项目描述

新能源汽车智能监控与分析系统主要涉及数据存储和数据管理 2 个方面，因此在云平台的构建设计上主要包含数据存储层和服务提供层。 本项目在 Linux 环境中（也可以在 Windows 操作系统中安装 Linux 虚拟机），以 Docker 容器为计算节点，构建分布式存储环境，在数据存储层通过 MySQL 数据库存储数据，在服务提供层通过 Tomcat 服务器提供应用服务。

知识目标

- 了解 Docker 容器技术。
- 了解基于 Docker 的私有云搭建技术。

技能目标

- 搭建云平台，并实现基础环境的部署。
- 在云平台上发布 Java Web 应用程序的。
- 在云平台上发布应用的镜像制作。

任务列表

任务编号	任务名称	建议课时
任务 10.1	安装 Linux 虚拟机	1
任务 10.2	安装与部署容器 Docker	1
任务 10.3	实现在云平台中部署 Tomcat 服务器	1
任务 10.4	实现在云平台中部署 MySQL 数据库	1
任务 10.5	实现在云平台中的项目部署	1
任务 10.6	制作镜像文件	1
技能训练	在虚拟机中搭建云平台	4
	总计：	10 课时

任务 10.1　安装 Linux 虚拟机

【任务描述】

安装 Linux 虚拟机可以让初学者放心大胆地进行操作和练习而不必担心因不熟练造成的宿主机问题。因此，本任务详细介绍了 Linux 虚拟机的安装过程，方便学生搭建虚拟环境，为接下来的私有云平台搭建做好准备。

微课 10
云平台部署发布

【任务目标】

知识目标

- 了解虚拟机的作用。

技能目标

- 能够正确安装与配置 Linux 虚拟机。

【任务实施】

(1) 配置系统基本环境

本次实验的环境为 CentOS 7 系统，在 Windows 中可以通过 VirtualBox 安装 CentOS 7 虚拟机进行实验。

通过 VirtualBox 安装 CentOS 7。

① 安装 VirtualBox。VirtualBox 的安装很简单，可一直按默认设置进行安装，不断地单击“下一步”按钮即可。中间如果遇到需要安装的驱动，选择安装。CentOS 7 的源使用 CentOS-7-x86_64-DVD-1511.iso。

② 创建 CentOS 7 虚拟机。打开 VirtualBox，如图 10-1-1 所示。单击左上方“新建”按钮，创建 CentOS 7 环境。

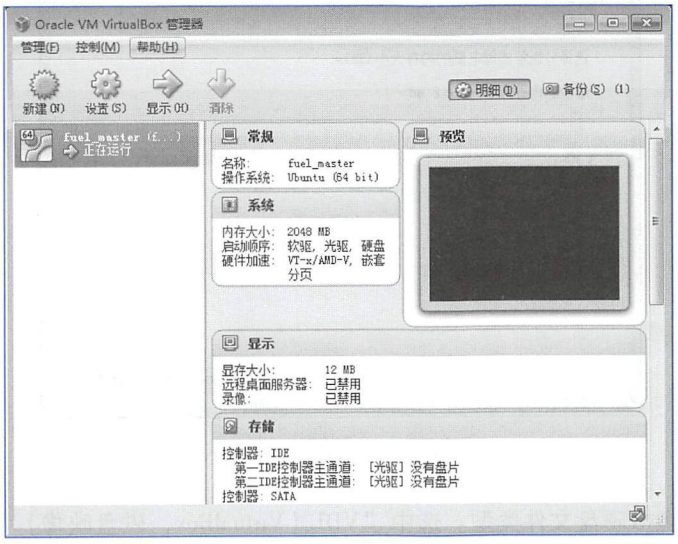


图 10-1-1　VirtualBox 主界面

③ 新建虚拟机环境。虚拟机名称为 CentOS 7，由于 VirtualBox 环境没有 CentOS 选项，所以在“版本”选项中，单击其右侧的下三角按钮▼，在弹出的列表中选择“Red Hat（64 Bit）”选项，如图 10-1-2 所示。完成设置后单击“下一步”按钮。

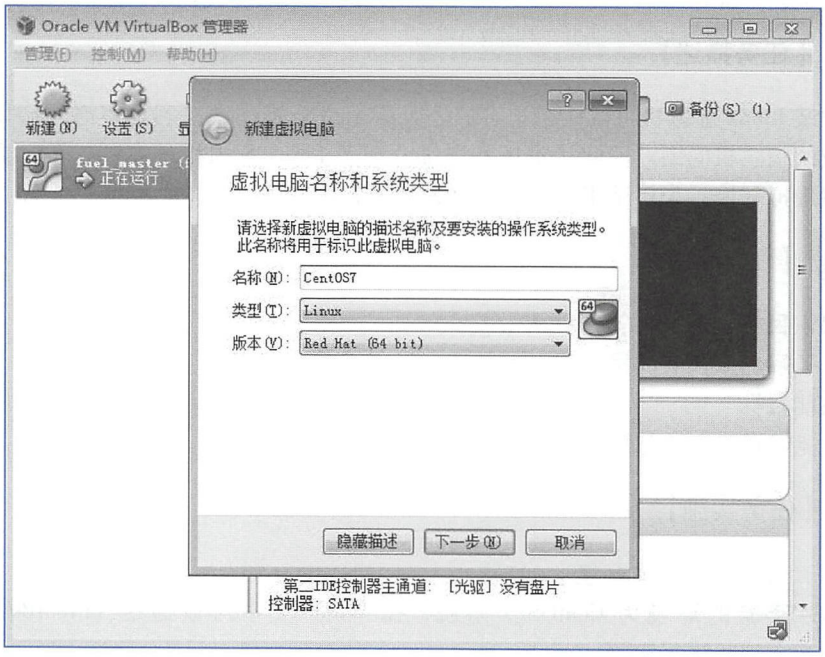


图 10-1-2　新建虚拟电脑

④ 分配内存大小。动态分配内存大小，如图 10-1-3 所示。这时用户可以拖动“滑块”来设置内存大小，建议内存空间设置大一点。完成后，单击“下一步”按钮。

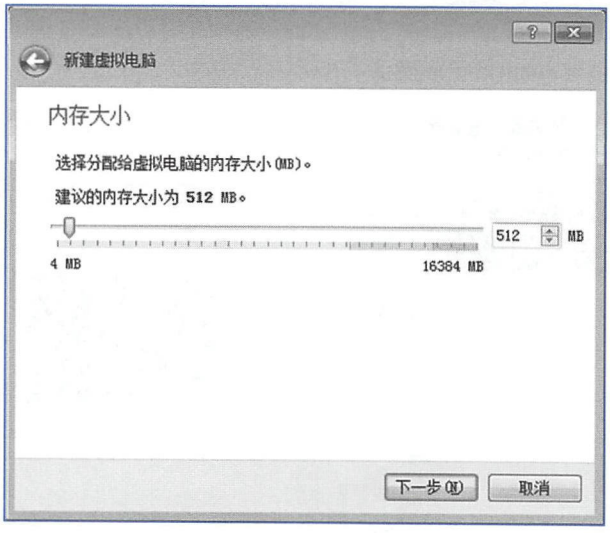


图 10-1-3　分配内存大小

⑤ 选择虚拟硬盘文件类型。选中“VDI（VirtualBox，磁盘映像）”单选按钮，设置虚拟硬盘文件类型为 VDI，如图 10-1-4 所示。完成后，单击“下一步”按钮。

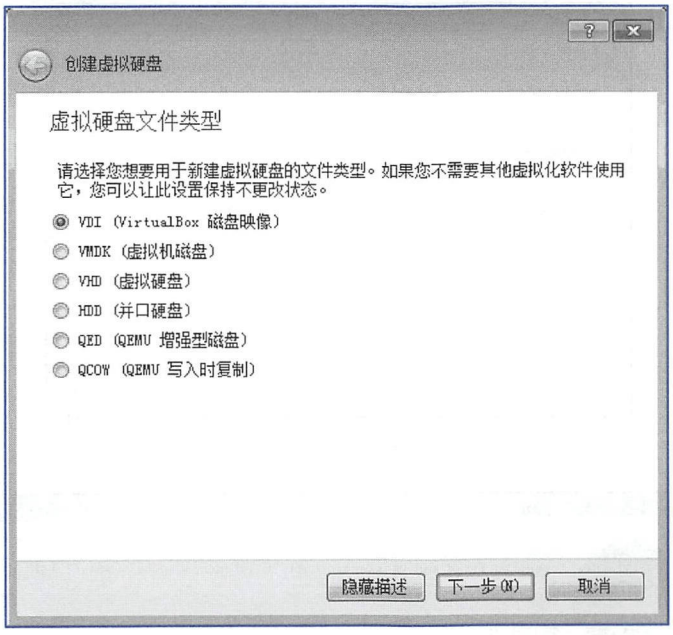


图 10-1-4　创建虚拟硬盘文件类型

⑥ 选择存储模式。选中“固定大小”单选按钮，设置存储在物理硬盘上的方式，如图 10-1-5 所示。完成后，单击“下一步”按钮。

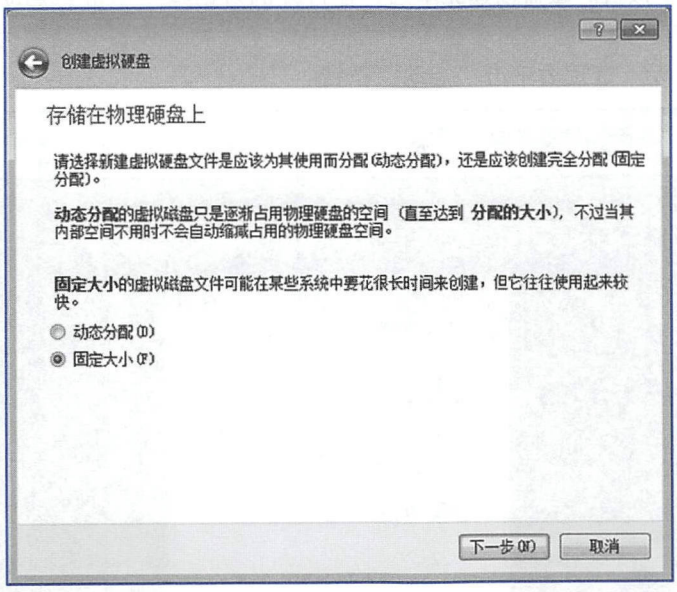


图 10-1-5　选择虚拟硬盘模式

⑦ 选择磁盘大小。拖动“滑块”，创建虚拟硬盘大小为 8 GB，如图 10-1-6 所示。然后单击“创建”按钮。

⑧ 创建磁盘。出现进度条，显示正在创建虚拟硬盘的进度，如图 10-1-7 所示。

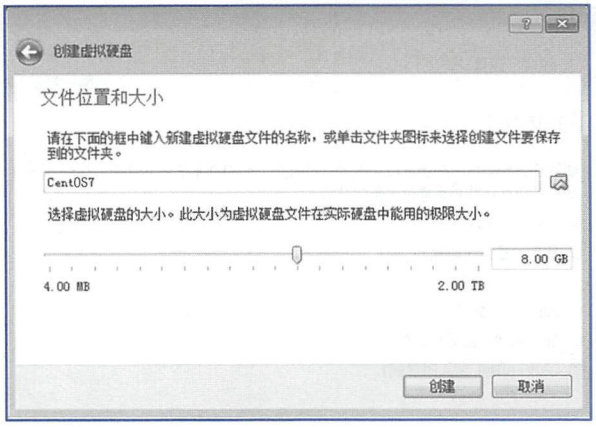


图 10-1-6　设置虚拟硬盘大小

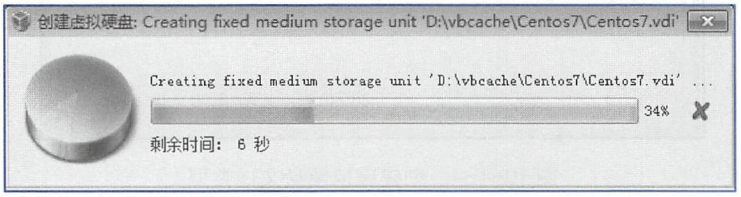


图 10-1-7　正在创建虚拟硬盘

(2) 安装虚拟机操作系统

① 启动虚拟机。磁盘创建完毕后，在打开的对话框中单击“启动虚拟机”按钮，然后在打开的“选择启动盘”对话框中选择“物理设备”选项为启动盘，如图 10-1-8 所示。选定后单击“启动”按钮。

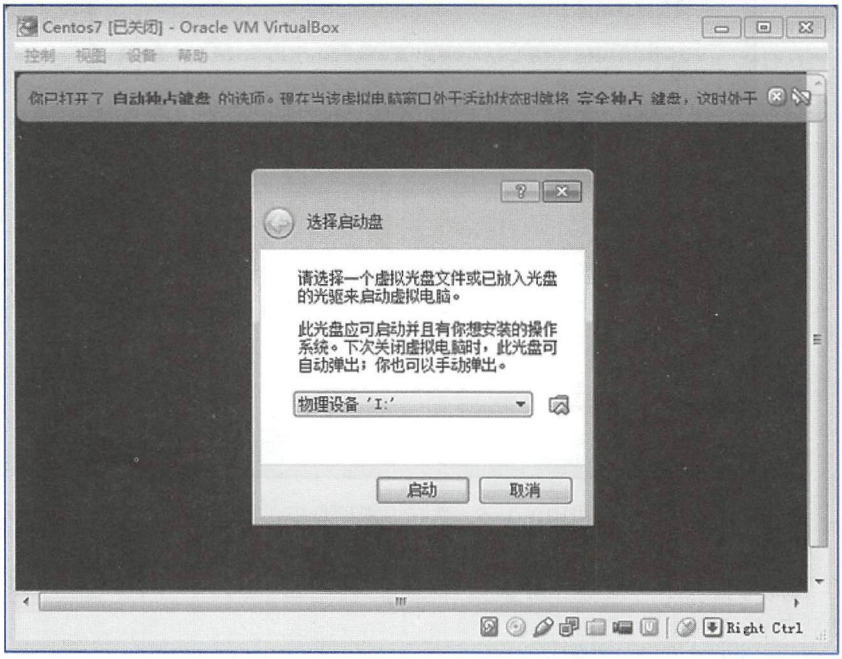


图 10-1-8　选择启动盘

② 系统安装。选择“Install CentOS 7”命令并按 Enter 键，如图 10-1-9 所示。

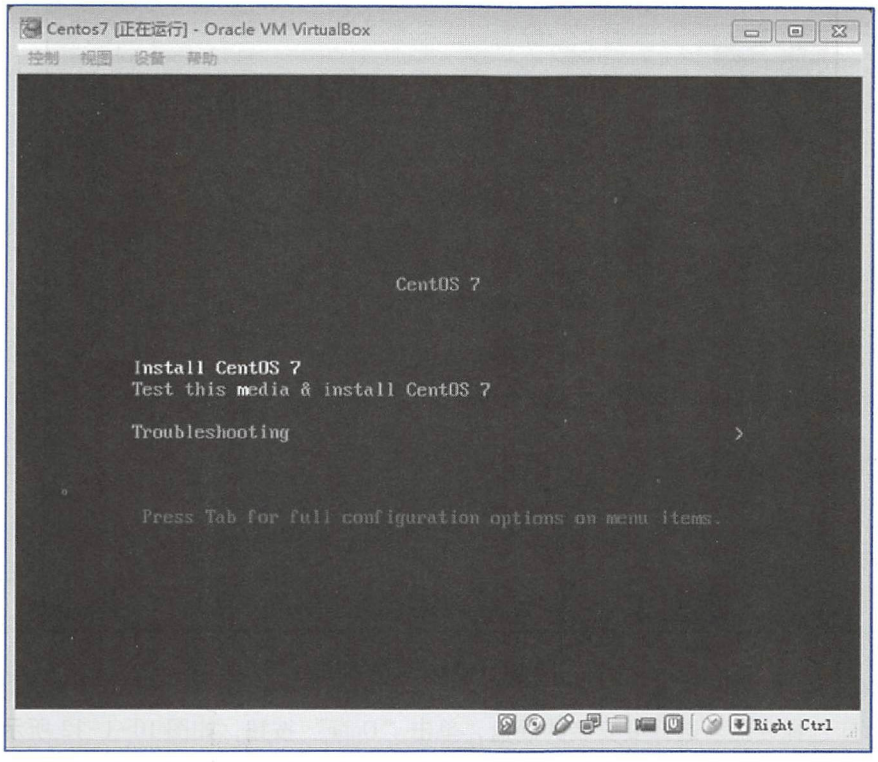


图 10-1-9　安装 CentOS 7

③ 安装语言升级。选择语言为“English→English（United States）”，如图 10-1-10 所示。选择完之后单击“Continue”按钮。

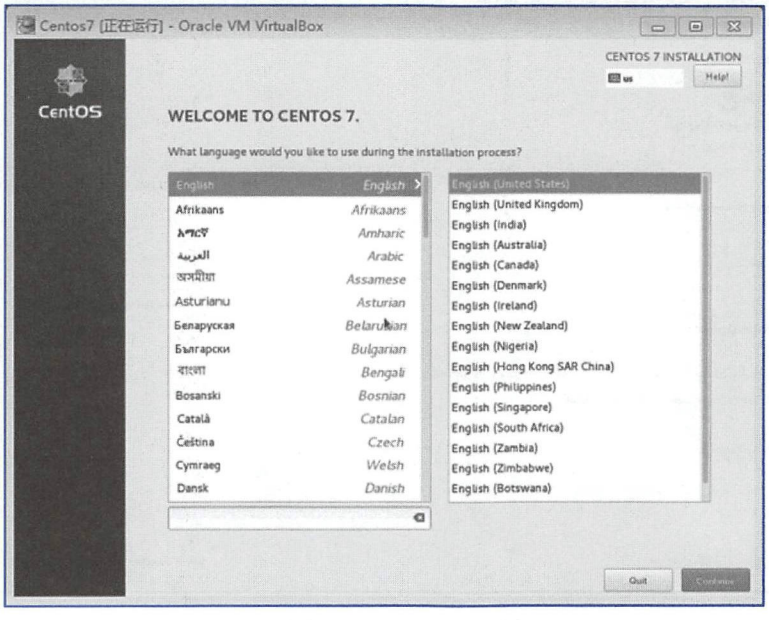


图 10-1-10　选择语言

④ 选择安装磁盘。单击“INSTALLATION SOURCE”按钮，如图 10-1-11 所示。

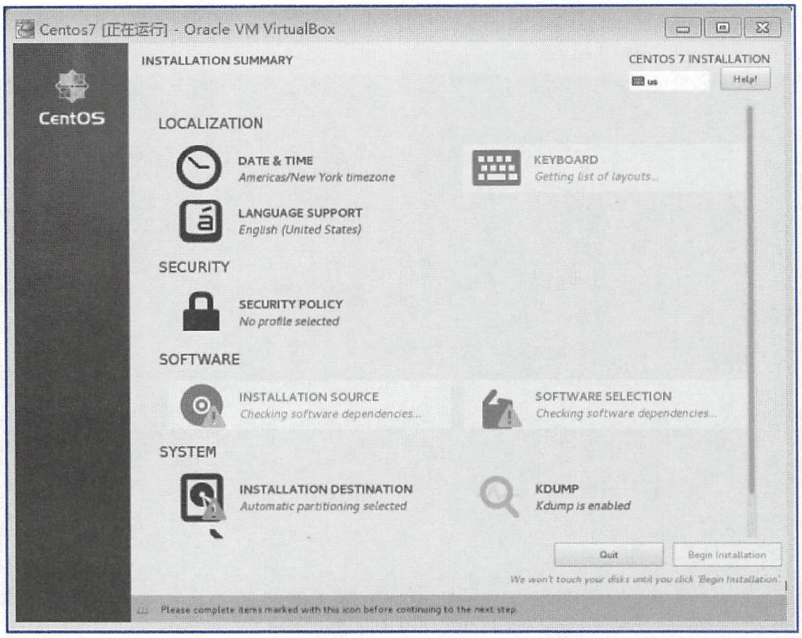


图 10-1-11　安装选项

⑤ 选择硬盘。选择要安装的硬盘，单击“Done”按钮，如图 10-1-12 所示。

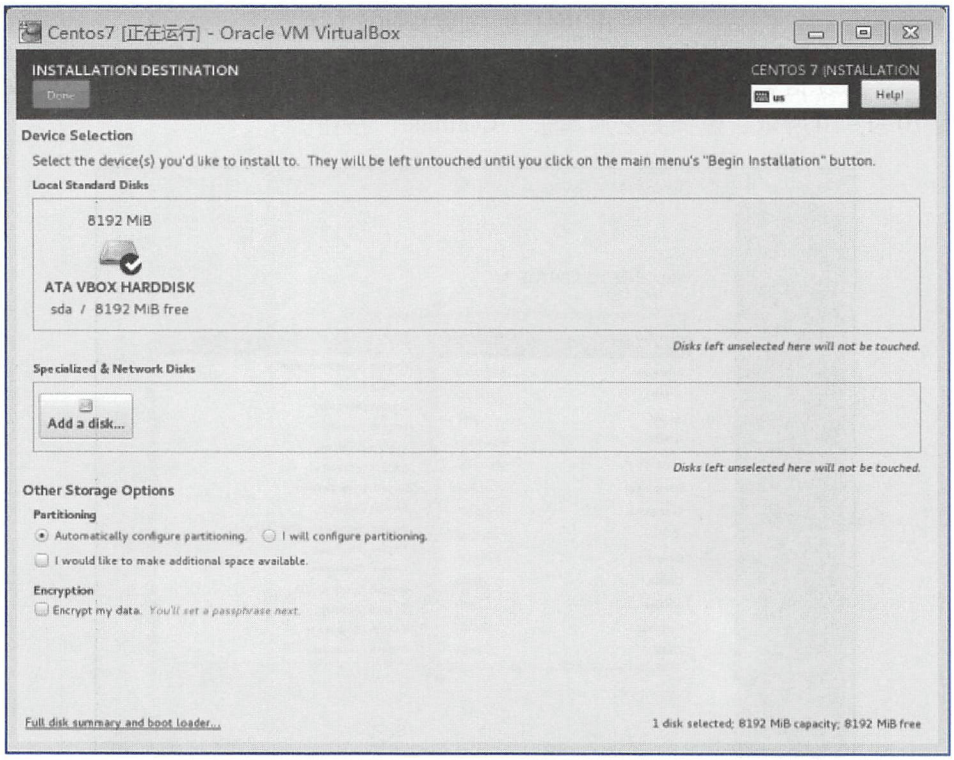


图 10-1-12　选择安装的硬盘

⑥ 系统安装。单击“Begin Installation”按钮，如图 10-1-13 所示。

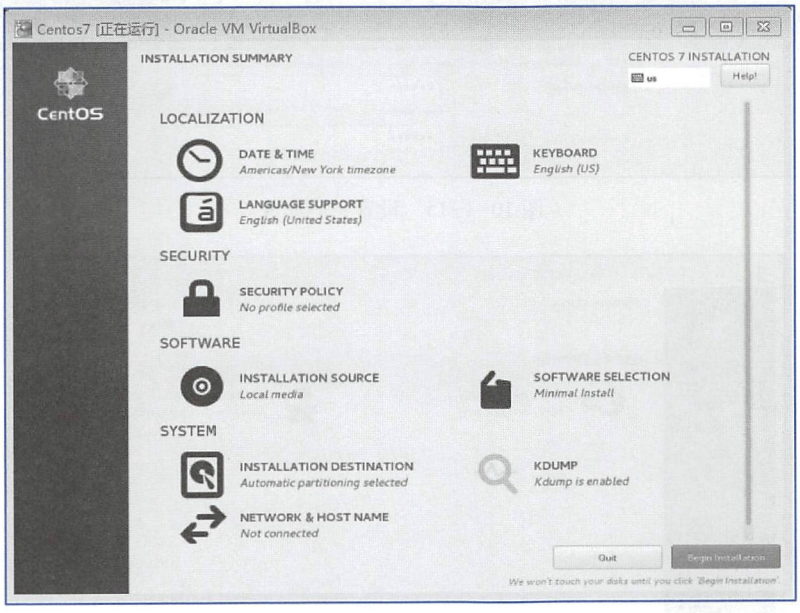


图 10-1-13　开始系统安装

⑦ 设置用户密码。单击“ROOT PASSWORD”按钮，进行设置密码，如图 10-1-14 所示。

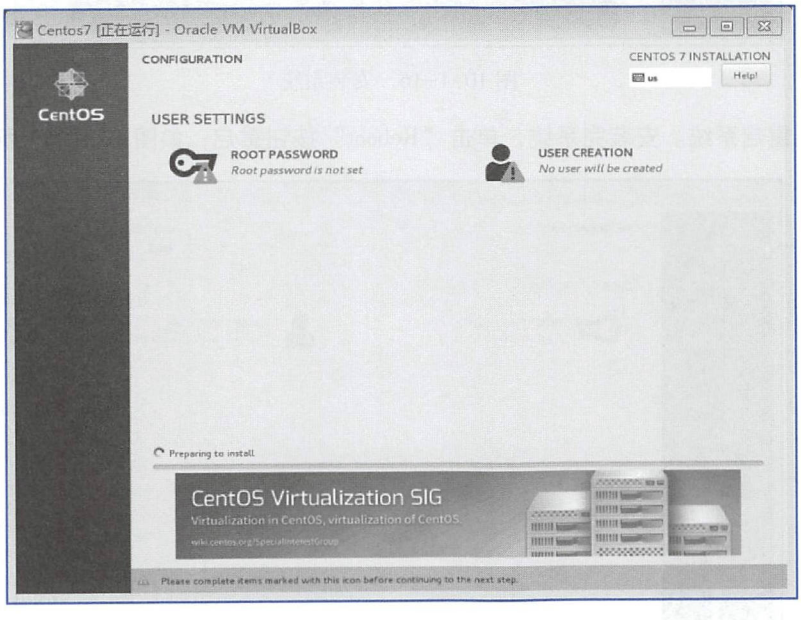


图 10-1-14　设置密码

在“Root Password”和“Confirm”文本框中输入“000000”，将密码设置为 000000，如图 10-1-15 所示。设置完后单击左上角“Done”按钮。

⑧ 安装系统。正在安装系统，如图 10-1-16 所示。

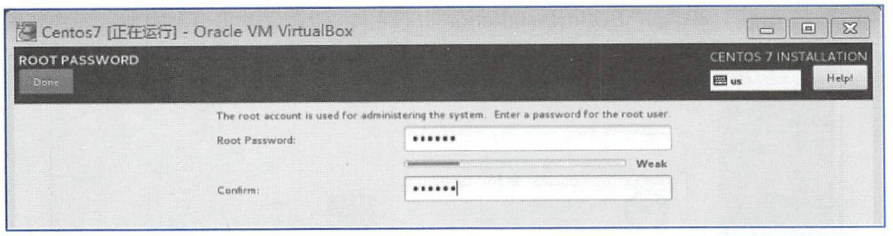


图 10-1-15　设置密码完成

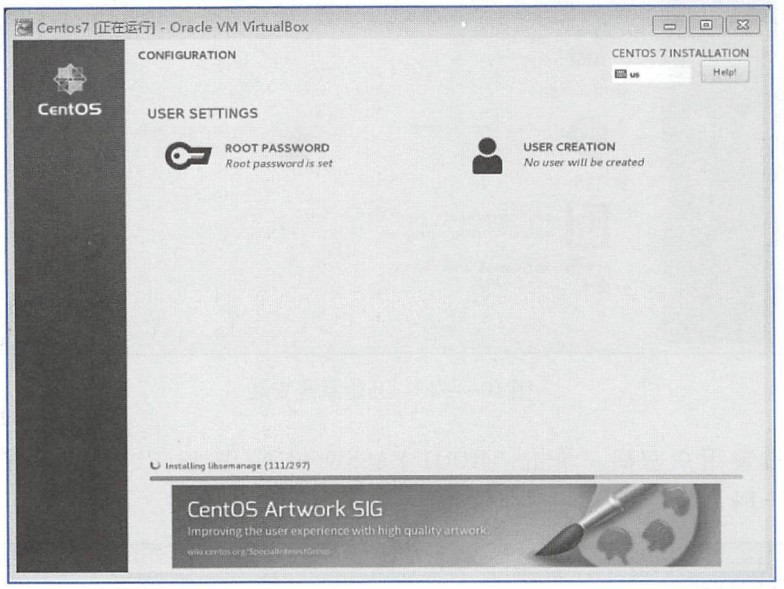


图 10-1-16　安装系统

⑨ 重启系统。安装完系统，单击“Reboot”按钮重启，如图 10-1-17 所示。

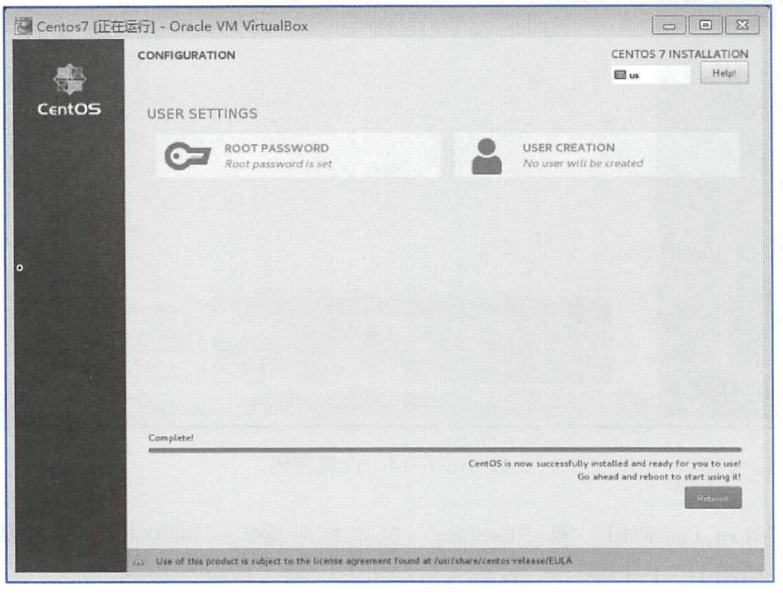


图 10-1-17　重启

（3）初始系统环境配置

登录 CentOS 7 系统，在“localhost login:”后光标所在处输入用户名“root”，然后按 Enter 键，系统执行完毕后，在光标所在处输入密码“000000”，如图 10-1-18 所示。

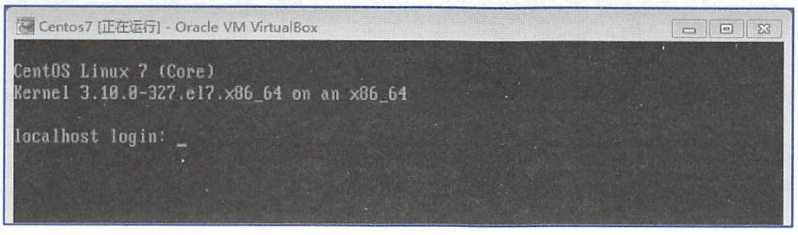


图 10-1-18　登录 CentOS 7 系统

任务 10.2　安装与部署 Docker 容器

【任务描述】

Docker 可以简化搭建私有云平台的过程，每个 Docker 容器内运行一个应用，不同的容器之间相互隔离，通过 Docker 可以虚拟出多个相互独立的应用服务。因此选择 Docker 作为搭建私有云平台的基础软件。Docker 安装过程没有图形化用户界面，全部都是通过命令行代码完成，本任务给出实施过程，关于 Linux 命令在此不做解释，读者可以通过网络资源自行了解相关命令。

【任务目标】

知识目标

- 了解 Docker 容器机制。

技能目标

- 能够正确安装与配置 Docker。

【任务实施】

（1）上传资源包

① 将 XianDian-PaaS-v2.1.iso 镜像文件传到系统中，本任务放到/opt 目录下，挂载镜像源，如图 10-2-1 所示。

② 通过命令行查看 opt 文件夹下的资源文件。

```
[root@k8s-test yum.repos.d]# cd /opt
[root@k8s-test opt]# ll
total 8740168
-rw-r--r-- 1 root root 8949929984 Apr 21 10:18 XianDian-PaaS-v2.1.iso
```

```
[root@k8s-test opt]# mount -o loop XianDian-PaaS-v2.1.iso /mnt
[root@k8s-test opt]# cd /mnt
[root@k8s-test mnt]# cp -rvf * /opt/
```

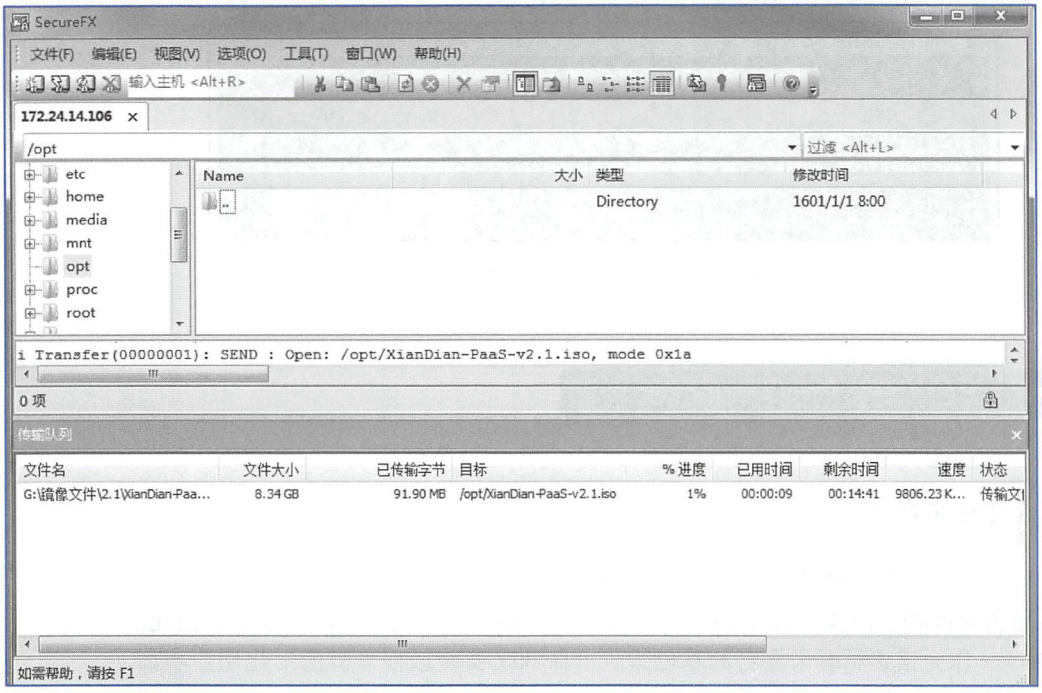


图 10-2-1　镜像文件上传

（2）配置 yum 源

① 编辑配置文件/etc/sysctl.conf，将以下内容添加在文件中。

```
# vi /etc/sysctl.conf
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
```

② 重新加载 sysctl.conf。

```
# sysctl -p
net.ipv4.ip_forward = 1
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
```

③ 删除 iptables 防火墙规则。

```
# iptables -F
# iptables -X
# iptables -Z
# /usr/sbin/iptables-save
```

④ 配置 yum 文件。

```
[root@k8s-test opt]# cd  /etc/yum.repos.d/
[root@k8s-test yum.repos.d]# ll
total 24
-rw-r--r--. 1 root root 1664 Mar 27  2015 CentOS-Base.repo
-rw-r--r--. 1 root root 1309 Mar 27  2015 CentOS-CR.repo
-rw-r--r--. 1 root root  649 Mar 27  2015 CentOS-Debuginfo.repo
-rw-r--r--. 1 root root  290 Mar 27  2015 CentOS-fasttrack.repo
-rw-r--r--. 1 root root 1331 Mar 27  2015 CentOS-Sources.repo
-rw-r--r--. 1 root root 1002 Mar 27  2015 CentOS-Vault.repo
[root@k8s-test yum.repos.d]# mv * /media/
[root@k8s-test yum.repos.d]# ll
total 0
[root@k8s-test yum.repos.d]# vi local.repo
[CentOS]
name=CentOS
baseurl=ftp://172.24.13.10/centos/
gpgcheck=0
enabled=1
```

⑤ 安装 ftp 服务。

```
[root@k8s-test yum.repos.d]# yum install -y vsftpd*
[root@k8s-test yum.repos.d]# vi /etc/vsftpd/vsftpd.conf
anon_root=/opt
# Example config file /etc/vsftpd/vsftpd.conf
[root@k8s-test yum.repos.d]# service vsftpd start
Redirecting to /bin/systemctl start  vsftpd.service
[root@k8s-test yum.repos.d]# chkconfig vsftpd on
```

⑥ 添加 docker 的 yum 源。

```
[root@k8s-test opt]# vi /etc/yum.repos.d/local.repo
[CentOS]
name=CentOS
baseurl=ftp://172.24.13.10/centos/  //IaaS 平台的 yum 源
gpgcheck=0
enabled=1
[paas]
name=paas
baseurl=ftp://172.24.14.106/docker  //Docker 的 yum 源
gpgcheck=0
enabled=1
```

(3) 安装 Docker 服务

① 安装 Docker 服务。

```
[root@k8s-test opt]# yum install -y docker
[root@k8s-test opt]# service docker start
[root@k8s-test opt]# chkconfig docker on
[root@k8s-test opt]# docker --version
Docker version 1.10.3, build cb079f6-unsupported
```

② 修改 Docker 配置文件。

```
[root@k8s-test docker_images]# vi /etc/sysconfig/docker
# /etc/sysconfig/docker
ADD_REGISTRY='--add-registry 172.24.14.106:5000' //修改成本地的私有仓库地址
INSECURE_REGISTRY='--insecure-registry 172.24.14.106:5000' //修改成本地的私有
                                                          //仓库地址
[root@k8s-test docker_images]# service docker restart
Redirecting to /bin/systemctl restart  docker.service
```

任务 10.3　实现在 Docker 中部署 Tomcat 服务器

【任务描述】

学生已经在技能训练篇中学习了如何在本地机（Windows 操作系统）安装配置 Tomcat，本任务重点讲解云平台中 Tomcat 的部署与运行。Docker 容器中 Tomcat 的安装，首先将资源包放置到/opt 文件夹下，将 Tomcat 上传至 Docker 私有仓库，启动 Tomcat 容器，通过浏览器访问该容器。

【任务目标】

知识目标

- 了解 Docker 中 Tomcat 容器的安装方法。

技能目标

- 在 Docker 中正确部署与配置 Tomcat 容器。

【任务实施】

① 上传 Tomcat 安装资源包。

利用 WinSCP 工具进行文件传输，把 tomcat-7.0:latest.tar 文件拷贝到/opt 目录下。

```
[root@k8s-test opt]# cd /opt
[root@k8s-test opt]# ll
total 9350192
dr-xr-xr-x 3 root root       4096 Oct 18 06:13 docker_images
-rw-r--r-- 1 root root    6418531 Oct 17 10:21 jenkins.zip
-rw-r--r-- 1 root root  618219520 Oct 20 02:31 tomcat-7.0:latest.tar
```

② 上传 tomcat-7.0：latest.tar 到私有仓库。

```
# mkdir /usr/jdk64/
[root@k8s-test opt]# docker load < tomcat-7.0:latest.tar
[root@k8s-test opt]# docker images
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
docker.io/bamos/openface      latest    7e537bef9341   4 months ago     2.543 GB
172.24.14.106:5000/mysql      latest    d9124e6c552f   11 months ago    383.4 MB
172.24.14.106:5000/registrylatest       c9bd19d022f6   12 months ago    33.27 MB
docker.io/registry            latest    c9bd19d022f6   12 months ago    33.27 MB
172.24.14.106:5000/ubuntu14.04.3        ebdc8e295a2e   21 months ago    187.9 MB
none    none 7c34bafd1150   2 years ago          601.2 MB
[root@k8s-test opt]# docker push 7c34bafd1150 172.24.14.106:5000/tomcat-7.0:latest
```

③ 启动 Tomcat 容器，查看 Tomcat 容器状态。

```
[root@k8s-test opt]# docker run -d -p 50080:8080 tomcat-7.0:latest
c50b67dcf596270c2390980f939b8444c437413862515325eea4926afeabeb19
[root@k8s-test opt]# docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
c50b67dcf596	docker.io/consol/tomcat-7.0:latest	"/bin/sh -c /opt/tomc"	10 seconds ago	Up 3 seconds	8778/tcp, 0.0.0.0:50080->8080/tcp	prickly_euler
99a262e66d53	mysql:latest	"docker-entrypoint.sh"	44 hours ago	Exited (0) 20 hours ago		gogs-mysql
2c67d4227586	bamos/openface	"./root/openface/demo"	2 days ago	Exited (0) 20 hours ago		big_perlman
5c2fe1ade791	registry:latest	"/entrypoint.sh /etc/"	2 days ago	Up About an hour	0.0.0.0:5000->5000/tcp	registry

注意

在 Windows 环境下 Tomcat 默认端口号是 8080，而在本任务中，启动 Tomcat 时指定其端口号是 50080。

④ 访问 172.24.14.106:50080 网址，查看 Tomcat 服务器状态。Tomcat 运行界面如图 10-3-1 所示。

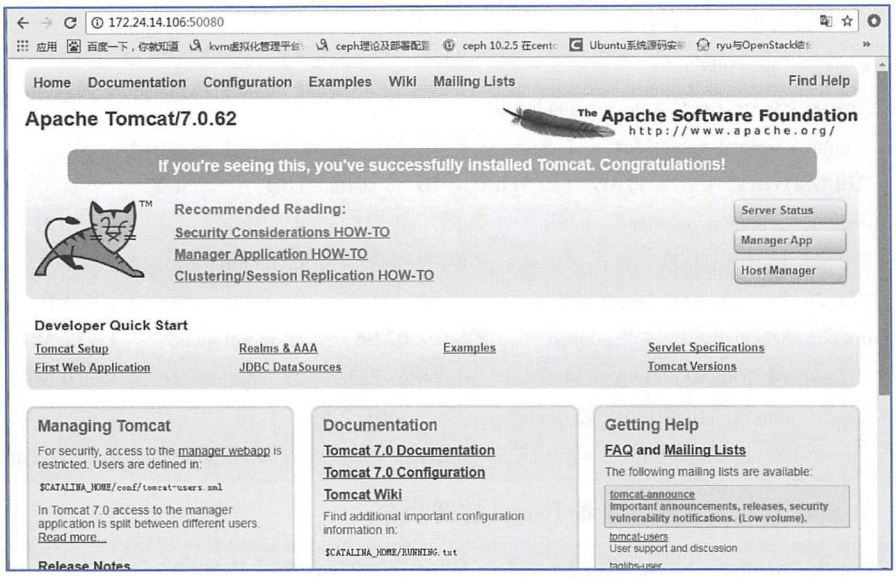


图 10-3-1　Tomcat 运行界面

任务 10.4　实现在 Docker 中部署 MySQL 数据库

【任务描述】

学生已经在技能训练篇中学习了如何在本地机（Windows 操作系统）安装配置 MySQL，本任务重点讲解云平台中 MySQL 的安装与运行。Docker 容器中 MySQL 的安装，首先将资源包放置到/opt 文件夹下，将 MySQL 上传至 Docker 私有仓库，启动 MySQL 容器。

【任务目标】

知识目标

- 了解 Docker 中 MySQL 容器的安装方法。

技能目标

- 在 Docker 中正确部署与配置 MySQL 容器。

【任务实施】

① 上传 MySQL 安装资源包。

利用 WinSCP 工具进行文件传输，把 mysql_latest. tar 文件拷贝到/opt 目录下。

② 上传 MySQL 镜像到私有仓库。

```
[root@k8s-test docker_images]# cd /opt/docker_images
[root@k8s-test docker_images]# ll
```

```
total 8692308
-r-xr-xr-x 1 root root 1323304960 Oct 17 10:19 mysql_latest.tar
-r-xr-xr-x 1 root root   33918976 Oct 17 10:20 registry_latest.tar
[root@k8s-test docker_images]# docker load < mysql_latest.tar
[root@k8s-test docker_images]# docker images
REPOSITORY      TAG             IMAGE ID            CREATED         SIZE
<none><none>d9124e6c552f        10 months ago   383.4 MB
172.24.14.106:latest   c9bd19d022f6        12 months ago   33.27 MB
5000/registry
docker.io/registry          latest      c9bd19d022f6        12 months ago 33.27 MB
172.24.14.106:14.04.3      ebdc8e295a2e        21 months ago 187.9 MB
5000/ubuntu
[root@k8s-test docker_images]# docker tag d9124e6c552f 172.24.14.106:5000/mysql:latest
[root@k8s-test docker_images]# docker push 172.24.14.106:5000/mysql:latest
The push refers to a repository [172.24.14.106:5000/mysql]
ee30b869dd90: Pushed
b5d824491b78: Pushed
b26238180bc8: Pushed
01e91410235e: Pushed
b610b16e919f: Pushed
1574ff8789b1: Pushed
e7048a1643a4: Pushed
1bc74a039df4: Pushed
6ebad06b3e49: Pushed
f1621398948b: Pushed
fe4c16cbf7a4: Pushed
latest: digest: sha256:3298591ea00ac8de8348aa63a2f8bba0a007844242cc8260efbde624363-
264a8 size: 2594
```

③ 拉取 MySQL 镜像到本地。

```
[root@k8s-test docker_images]# docker pull mysql:latest
Trying to pull repository 172.24.14.106:5000/mysql ...
latest: Pulling from 172.24.14.106:5000/mysql
Digest: sha256:3298591ea00ac8de8348aa63a2f8bba0a007844242cc8260efbde624363264a8
Status: Image is up to date for 172.24.14.106:5000/mysql:latest
```

④ 启动容器。

```
[root@k8s-test docker_images]#
# docker run -d -p 13306:3306 -e MYSQL_ROOT_PASSWORD=000000 --name gogs-
mysql docker.io/mysql:latest
25d354ad8947661043fc5a9c54520a5dfc5c5c197a827f626ab7af838b70ff61
```

端口映射：13306：3306。13306 为外部端口，3306 为容器内部端口。

⑤ 查看启动效果。

```
# docker ps -a
CONTAINER ID IMAGE    COMMAND    CREATED      STATUS
25d354ad8947      mysql:latest "docker-entrypoint. sh"    7 seconds ago    Up 5 seconds
PORTNAMES
0. 0. 0. 0:13306->3306/tcp          gogs-mysql
```

任务 10. 5　实现在云平台中的项目部署

【任务描述】

将本书中所涉及的项目“新能源汽车智能监控与分析系统”发布到云平台中的 Tomcat 服务器。将项目先打成 war 包，然后上传至 Tomcat 容器的指定目录，启动 Tomcat 服务器即可。

【任务目标】

知识目标

- 了解 Docker 中 Tomcat 容器的项目部署方法。

技能目标

- 在 Docker 中正确将项目部署到 Tomcat 容器。

【任务实施】

① 修改程序中数据库访问的端口号为 13306，并将工程打成 war 包。

② 将工程的 war 包拷贝到 Tomcat 服务器中。

```
docker cp /newcar. war prickly_euler:/opt/apache-tomcat-7. 0. 62/webapps/newcar. war
```

③ 启动 Tomcat 服务器。

```
[root@ k8s-test /]# docker exec -it c50b67dcf596 bash
root@ c50b67dcf596:/# cd /opt/apache-tomcat-7. 0. 62/bin/
bootstrap. jar                configtest. bat                setclasspath. bat
tomcat-juli. jar
catalina-tasks. xml           configtest. sh                 setclasspath. sh
tomcat-native. tar. gz
catalina. bat                 daemon. sh                     shutdown. bat
tool-wrapper. bat
catalina. sh                  deploy-and-run. sh             shutdown. sh
tool-wrapper. sh
commons-daemon-native. tar. gz  digest. bat                  startup. bat
version. bat
```

```
commons-daemon.jar          digest.sh                    startup.sh
version.sh
# chmod +x *.sh
# ./shutdown.sh
# ./startup.sh
Using CATALINA_BASE:   /opt/apache-tomcat-7.0.56
Using CATALINA_HOME:   /opt/apache-tomcat-7.0.56
Using CATALINA_TMPDIR: /opt/apache-tomcat-7.0.56/temp
Using JRE_HOME:        /usr/jdk64/jdk1.8.0_77
Using CLASSPATH:
/opt/apache-tomcat-7.0.56/bin/bootstrap.jar:/opt/apache-tomcat-7.0.56/bin/tomcat-juli.jar
Tomcat started.
```

④ 查看启动日志。

```
# tail -f ../logs/catalina.out
Jun 30, 2017 10:42:09 AM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Jun 30, 2017 10:42:09 AM hudson.WebAppMain$3 run
INFO: necar.war is fully up and running
Jun 30, 2017 10:42:10 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tasks.Maven.MavenInstaller
Jun 30, 2017 10:42:12 AM hudson.model.DownloadService$Downloadable load
INFO: Obtained the updated data file for hudson.tools.JDKInstaller
Jun 30, 2017 10:42:12 AM hudson.model.AsyncPeriodicWork$1 run
INFO: Finished Download metadata. 10,377 ms
```

任务 10.6　制作镜像文件

【任务描述】

项目在云平台部署好后，将整个运行环境制作成镜像文件，极大地方便了测试工程师的测试工作，有效地提高了工作效率。制作镜像文件之前，需要查看 Tomcat 容器的镜像编号，再通过 docker export 命令将对应编号的容器制作成镜像文件。

【任务目标】

知识目标

- 了解 Docker 中镜像文件的制作方法。

技能目标

- 制作 Docker 镜像文件。

【任务实施】

① 查看部署好环境的容器 ID。

```
[root@k8s-test opt]# docker ps -a
CONTAINER ID   IMAGE                    COMMAND                  CREATED          STATUS          PORTS                                                  NAMES
c50b67dcf596   docker.io/consol/tomcat-7.0:latest   "/bin/sh -c /opt/tomc"   42 minutes ago   Up 7 minutes    8778/tcp, 0.0.0.0:50080->8080/tcp                      prickly_euler
99a262e66d53   mysql:latest             "docker-entrypoint.sh"   45 hours ago     Up 36 minutes   0.0.0.0:13306->3306/tcp                                gogs-mysql
2c67d4227586   bamos/openface           "./root/openface/demo"   2 days ago       Up 36 minutes   0.0.0.0:8000->8000/tcp, 0.0.0.0:9000->9000/tcp         big_perlman
5c2fe1ade791   registry:latest          "/entry-point.sh /etc/"  2 days ago       Up 2 hours      0.0.0.0:5000->5000/tcp                                 registry
```

② 将容器保存成镜像。

```
[root@k8s-test opt]# docker export c50b67dcf596 > newcar:v1.tar
[root@k8s-test opt]# ll
total 10092008
dr-xr-xr-x 3 root root          4096 Oct 17 10:18 docker
dr-xr-xr-x 3 root root          4096 Oct 18 06:13 docker_images
-rw-r--r-- 1 root root  759619072 Oct 20 03:15 newcar:v1.tar
-rw-r--r-- 1 root root  618219520 Oct 20 02:31 tomcat-7.0:latest.tar
-rw-r--r-- 1 root root 8949929984 Apr 21 10:18 XianDian-PaaS-v2.1.iso
drwxr-xr-x 7 root root          136 Oct 19 06:06 yum
[root@k8s-test opt]#
```

制作好的镜像文件可以通过 Docker 导入、启动即可运行，至此一个 Java Web 项目就可以正常地在基于 Docker 的私有云平台运行了。

技能训练

在虚拟机中搭建云平台

任务描述

请学生根据本项目讲解的步骤在个人电脑搭建私有云平台。

任务分析

搭建私有云平台分为如下步骤：

① 如果是 Linux 操作系统可以直接进行，如果是 Windows 操作系统需先安装 Linux 虚拟机，参照任务 10.1。

② 在 Linux 环境下安装 Docker 容器，参照任务 10.2。

③ 在 Docker 中分别部署 Tomcat 和 MySQL 数据库，参照任务 10.3 和任务 10.4。

④ 将已经开发好的项目工程 war 包发布到 Tomcat 中，参照任务 10.5。

⑤ 制作镜像文件，参照任务 10.6。

项目总结

本项目讲解了一个简单的基于 Docker 私有云平台的搭建过程，在这个云平台中虚拟出数据存储层和服务提供层 2 个相对独立的应用容器，将“技能训练篇”中完成的项目发布在容器中，并将其制作成镜像文件，极大地方便了测试和运维这 2 项典型工作，整个过程涉及了目前比较流行的软件开发和维护过程。

参考文献

[1] 顾炯炯．云计算架构技术与实践［M］. 2 版．北京：清华大学出版社，2016.
[2] 刘鹏．云计算［M］. 北京：电子工业出版社，2015.
[3] 杨超伟，黄群英．空间云计算——应用与实践［M］. 北京：高等教育出版社，2015.
[4] 浙江大学软件工程实验室．Docker 容器与容器云［M］. 2 版．北京：人民邮电出版社，2016.
[5] 贺臣．Bootstrap 基础教程［M］. 北京：电子工业出版社，2016.
[6] 韩路彪．看透 Spring MVC：源代码分析与实践［M］. 北京：机械工业出版社，2015.
[7] 刘增辉．MyBatis 从入门到精通［M］. 北京：电子工业出版社，2017.

CETC 云计算技术与应用专业校企合作系列教材

- Android云存储客户端开发
- 软件定义网络（SDN）技术与应用
- Docker容器技术与应用
- 云存储技术与应用
- Hadoop大数据平台构建与应用
- 云计算数据中心运维
- 虚拟化技术与应用
- 云计算网络技术与应用
- Python语言
- ✓ Java Web云应用开发项目式教程

ISBN 978-7-04-049836-3

9 787040 498363

定价 43.60 元